PV - Wirkungsgrad

Contents

Geladen werden die Daten, die von der PV-Anlage mit Hilfe des SMA-Portals gewonnen werden. Ziel ist es, in der Tabelle *data* alle Datensätze aus den SMA-Daten, ergänzt um Hilfsgrößen, zur Verfügung zu stellen.

Die Daten liegen in Dateien tageweise vor, beim Download werden diese von Hand benannt, sie enthalten Datensätze (Zeilen), die im 5-Minuten-Rhythmus erfasst wurden. Beim Einlesen werden sie zusammengefügt.

Die Datensätze enthalten die Größen

leistung.pv — leistung.stp — netzeinspeisung — netzbezug — batt_ladung — batt_entladung — ladezustand

Die beiden ersten Werte sind identisch, deswegen wird "leistung.stp"" in der Folge sofort gelöscht.

1 Vorbereitungen

1.0.1 Laden der nötigen Bibliotheken.

```
source("01-Bibliotheken-laden.R")
```

1.0.2 Zur Auswertung werden einige Funktionen benötigt, die hier definiert werden.

```
source("02-Funktionen-bilden.R")
```

1.0.3 Einlesen der Dateien "Daten_dd_mm_yyyy.csv":

Alle Größen in der Einheit W, mit Ausnahme von 'ladezustand', dieser wird beim Lesen als Prozentsatz übergeben und anschließend auf 10000 = 100% normiert weil die Batterie eine Kapazität von annähernd $10kWh$ besitzt kann dies auch als Wh gelesen werden. Die Zeilen müssen sortiert werden, weil die Dateien nicht in der korrekten zeitlichen Reihenfolge eingelesen werden.

```
# Einlesen der Datenfiles-----notig: 02-Funktionen-bilden.R----------------------
source("03-Files-einlesen.R")
```

```
## gelesen:
##  Daten_01_10_2017.csv Daten_01_11_2017.csv Daten_01_12_2017.csv Daten_02_10_2017.csv Daten_02_11_201
```

1.0.4 Einige Spalten werden erzeugt, gelöscht, bearbeitet und z.B. neu normiert:

a) Die neue Spalte 'ct' zaehlt die Datenzeilen
b) Über 'ladezustand' laeuft eine Glaettungsfunktion, um einzelne Ausfaelle in den Messungen zu beseitigen.
c) 'month', 'day' und 'hour' werden aus der Variablen 'zeit' extrahiert und im Datumsformat "yyyy-mm-dd" bzw. als Zahl 0 - 23 gespeichert.
d) 'ladediff' wird als Differenz von Ladezustand zwischen dem aktuellen Zustand und dem vorangegangenen berechnet (Einheit Wh).
e) 'batt_ladung' und 'bat_entladung' werden von W in Wh umgerechnet (W in der Zeit 5 min, deswegen Division durch 12). Anm.: In der späteren Auswertung wird dies so interpretiert: Eine zur Zeit t erbrachte Leistung P führt zu einer el. Arbeit von $P \cdot 5\text{min}$ im Zeitintervall $t \pm 2,5\text{min}$

```
# Ergaenzende Spaltenoperationen ------------------------------------------------
source("03-Spalten-bearbeiten.R")
```

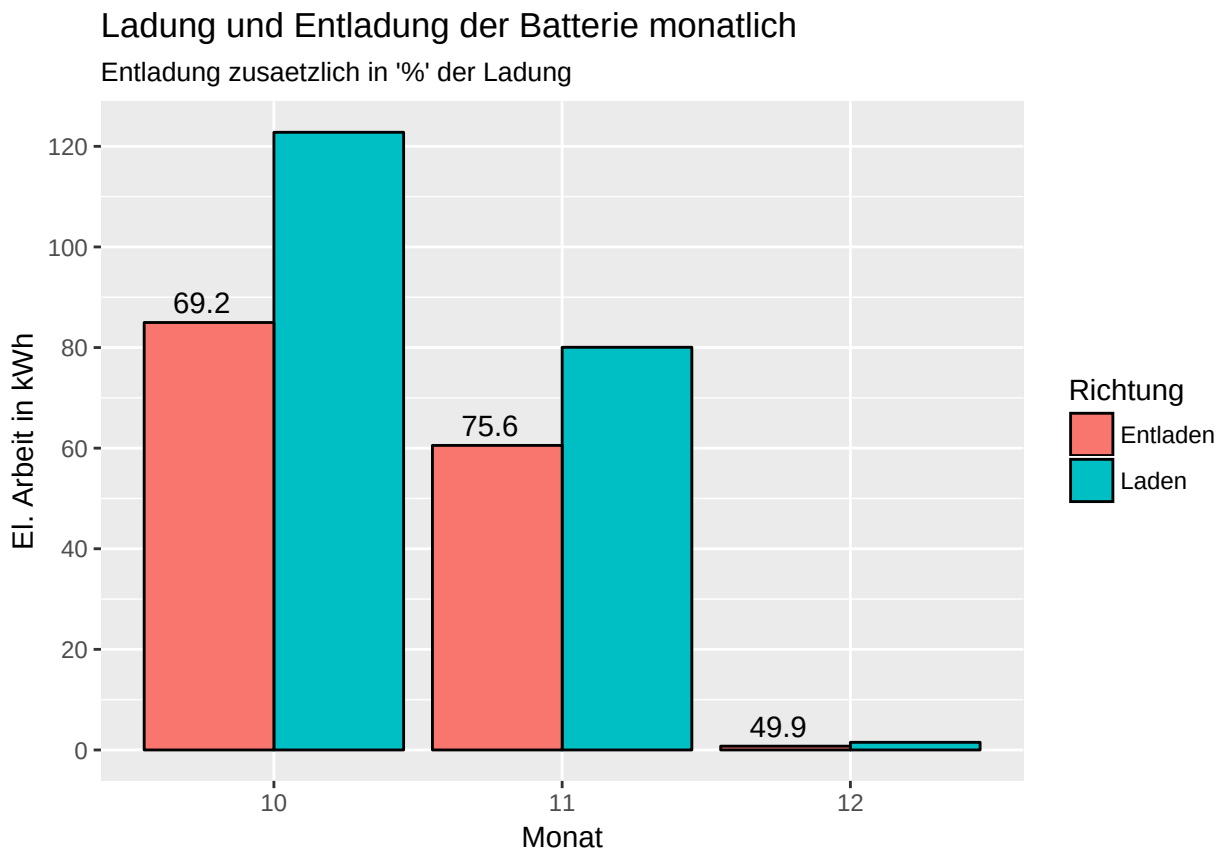
```
## Erzeuge Tabelle Verbrauch
## Loesche Spalten aus data leistung.pv leistung.stp netzeinspeisung netzbezug .
## Der Datensatz enthaelt jetzt  17849 Zeilen.
```

2 Auswertungen

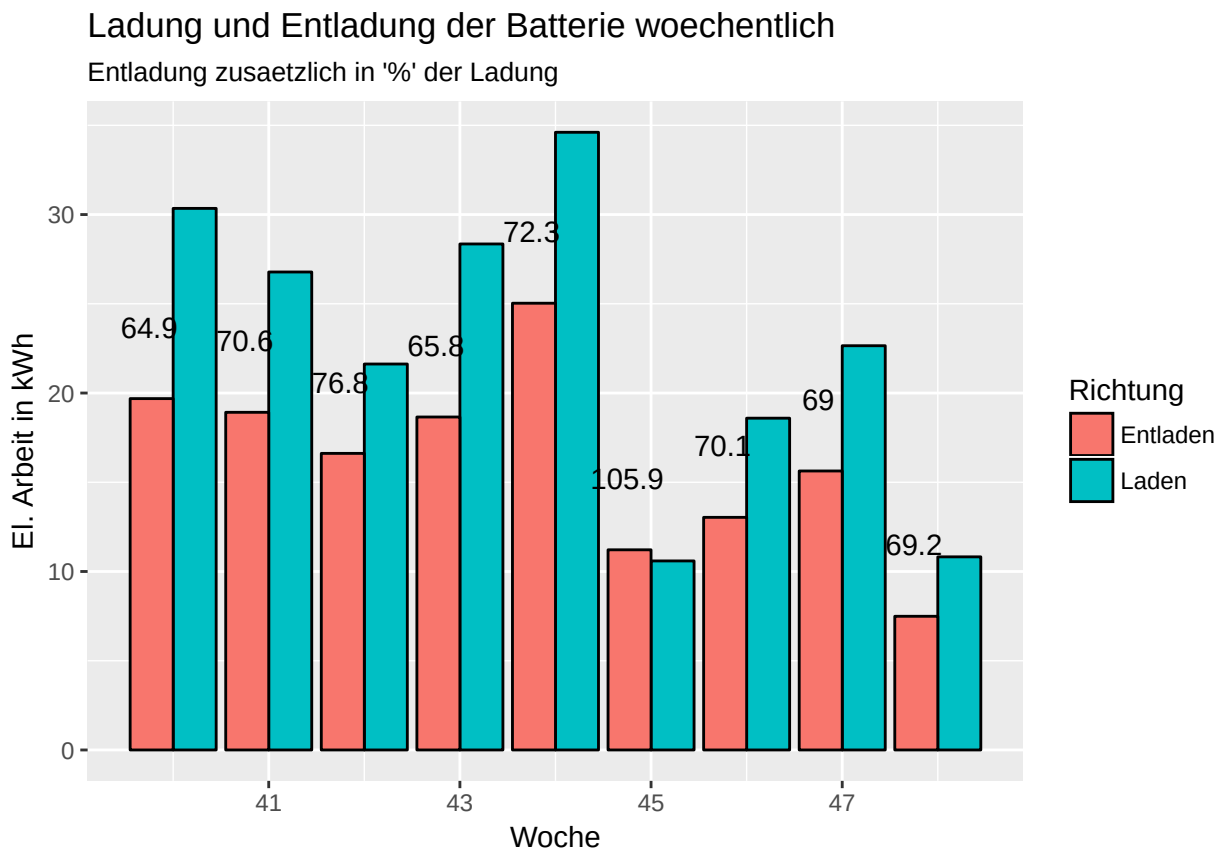
2.1 Einfache Summenbildungen

2.1.1 Monatssummen

```
source("04_Monatliche_Batterie_Ladung.R", print.eval=TRUE)
```



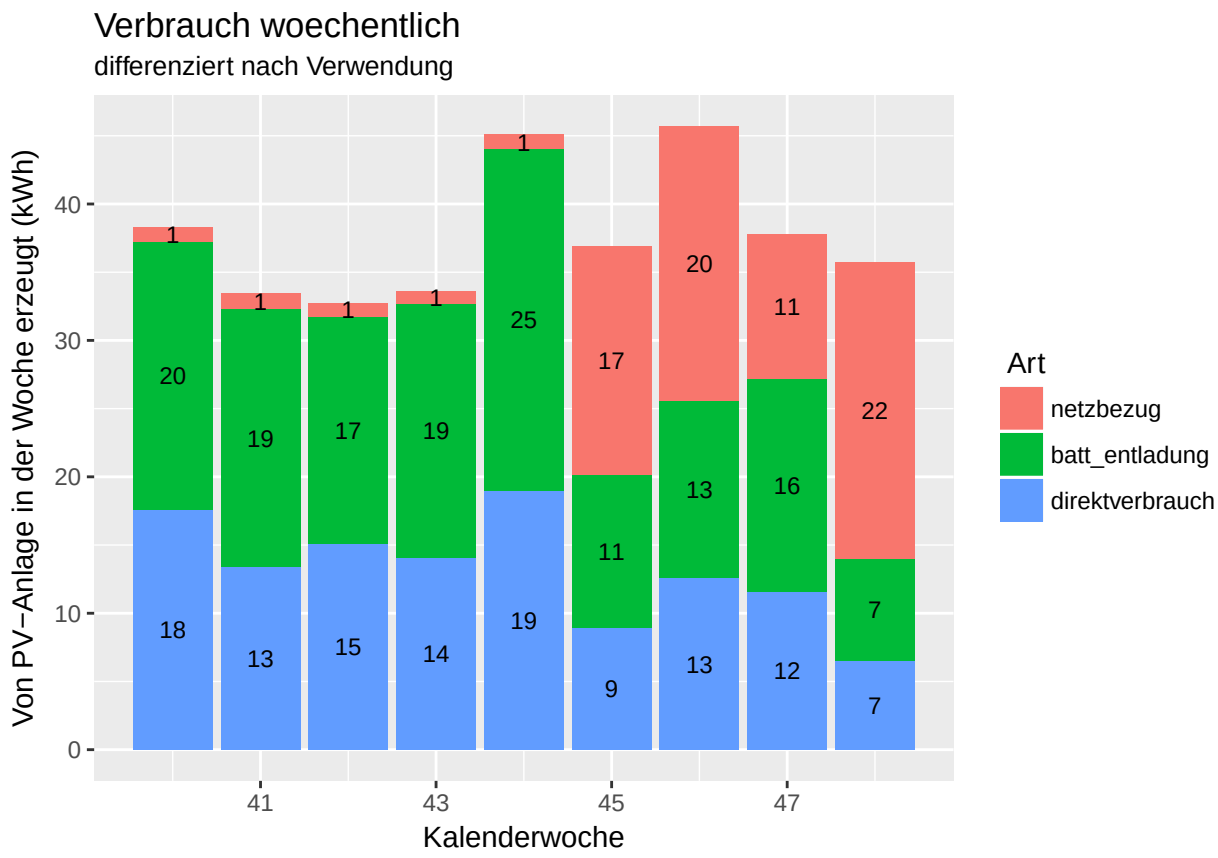

```
source("04_Woechentliche_Batterie_Ladung.R", print.eval=TRUE)
```



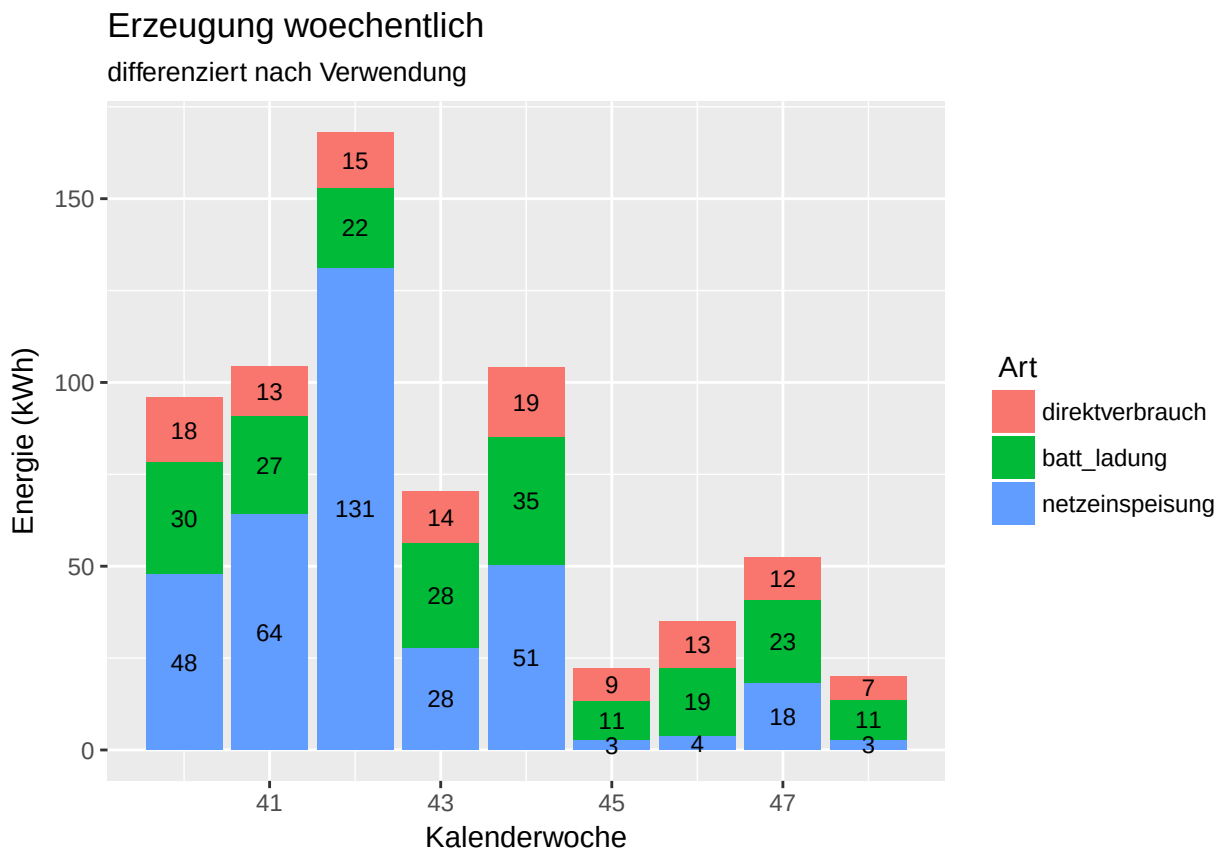

```
#ggsave("Wochen_Wirkungsg.pdf")
```

2.1.2 Eigenverbrauch - auch fürs Finanzamt

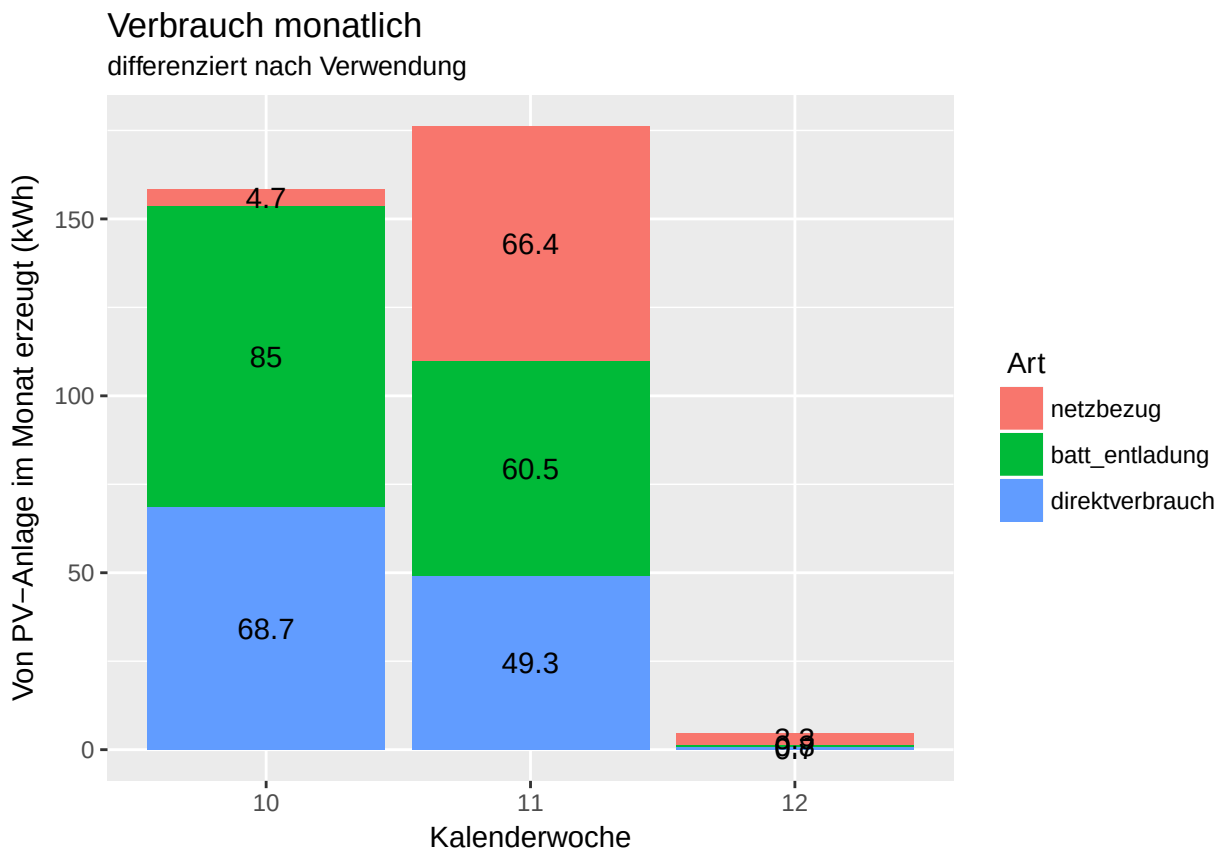
```
zusfg <- verbrauch %>%
         group_by(week) %>%
         summarise(batt_ladung     = sum(batt_ladung)/1000,
                   batt_entladung  = sum(batt_entladung)/1000,
                   leistung.pv     = sum(leistung.pv)/1000,
                   netzeinspeisung = sum(netzeinspeisung)/1000,
                   netzbezug       = sum(netzbezug)/1000) %>%
         mutate(   direktverbrauch = leistung.pv - netzeinspeisung - batt_ladung,
                   eigenverbrauch  = batt_entladung + direktverbrauch
                   )

zusfg.long <- melt(zusfg , id = "week", measure = c("netzbezug","batt_entladung", "direktverbrauch"))
ggplot(zusfg.long, aes(week, value, fill = variable)) +
   geom_bar(stat = "identity", position = "stack") +
   labs(     x  = "Kalenderwoche",
             y  = "Von PV-Anlage in der Woche erzeugt (kWh)",
          fill  = " Art",
          title = "Verbrauch woechentlich",
          subtitle = "differenziert nach Verwendung") +
   geom_text(aes(label=round(value,-0)), size= 3, data = zusfg.long, position = position_stack(vjust=0.5
```



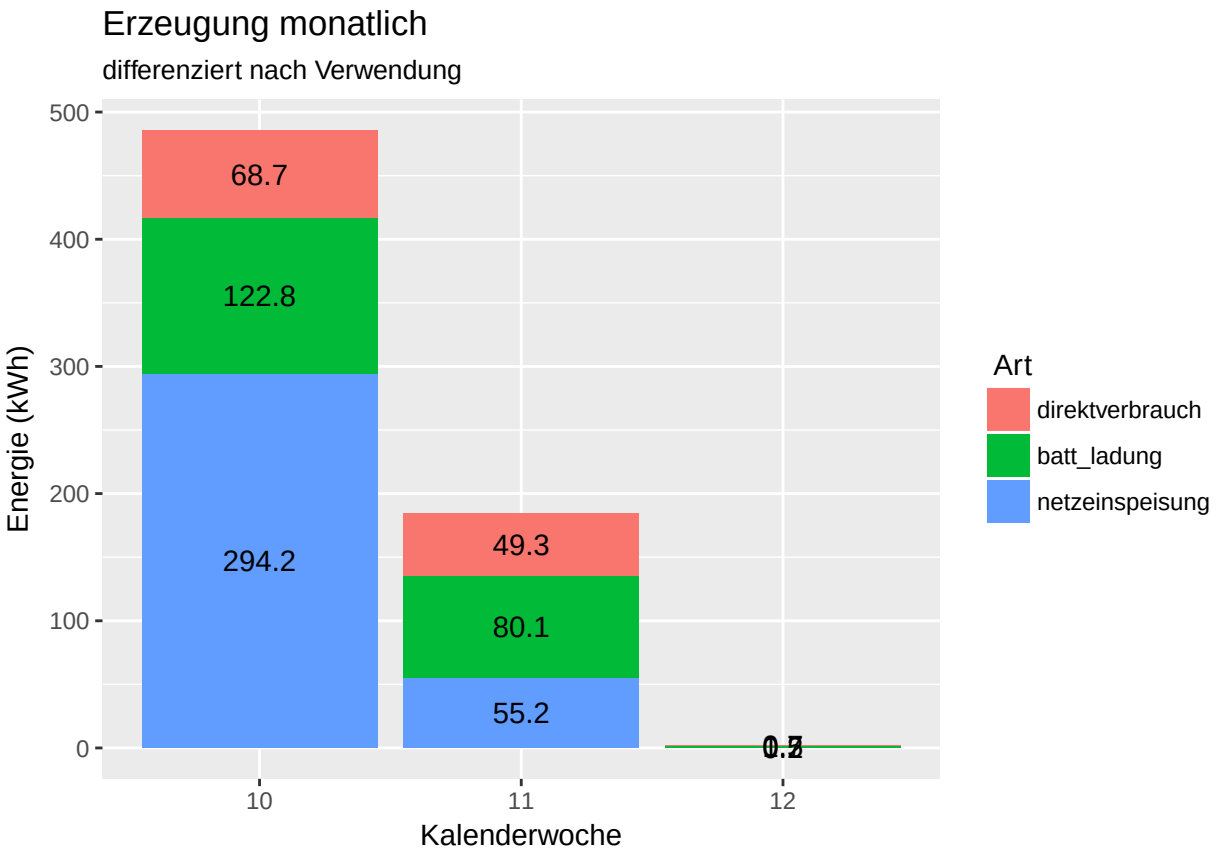

```
zusfg.long <- melt(zusfg , id = "week", measure = c("direktverbrauch","batt_ladung","netzeinspeisung" ))
ggplot(zusfg.long, aes(week, value, fill = variable)) +
   geom_bar(stat = "identity", position = "stack") +
   labs(     x  = "Kalenderwoche",
             y  = "Energie (kWh)",
          fill  = " Art",
          title = "Erzeugung woechentlich",
          subtitle = "differenziert nach Verwendung") +
   geom_text(aes(label=round(value,-0)), size= 3, data = zusfg.long, position = position_stack(vjust=0.5
```




```
zusfg <- verbrauch %>%
         group_by(month) %>%
         summarise(batt_ladung     = sum(batt_ladung)/1000,
                   batt_entladung  = sum(batt_entladung)/1000,
                   leistung.pv     = sum(leistung.pv)/1000,
                   netzeinspeisung = sum(netzeinspeisung)/1000,
                   netzbezug       = sum(netzbezug)/1000) %>%
         mutate(   direktverbrauch = leistung.pv - netzeinspeisung - batt_ladung,
                   eigenverbrauch  = batt_entladung + direktverbrauch
                   )

zusfg.long <- melt(zusfg , id = "month", measure = c( "netzbezug", "batt_entladung", "direktverbrauch"))

ggplot(zusfg.long, aes(month, value, fill = variable)) +
   geom_bar(stat = "identity", position = "stack") +
   labs(     x  = "Kalenderwoche",
             y  = "Von PV-Anlage im Monat erzeugt (kWh)",
          fill  = " Art",
          title = "Verbrauch monatlich",
          subtitle = "differenziert nach Verwendung") +
   geom_text(aes(label=round(value,1)), data = zusfg.long, position = position_stack(vjust=0.5))
```

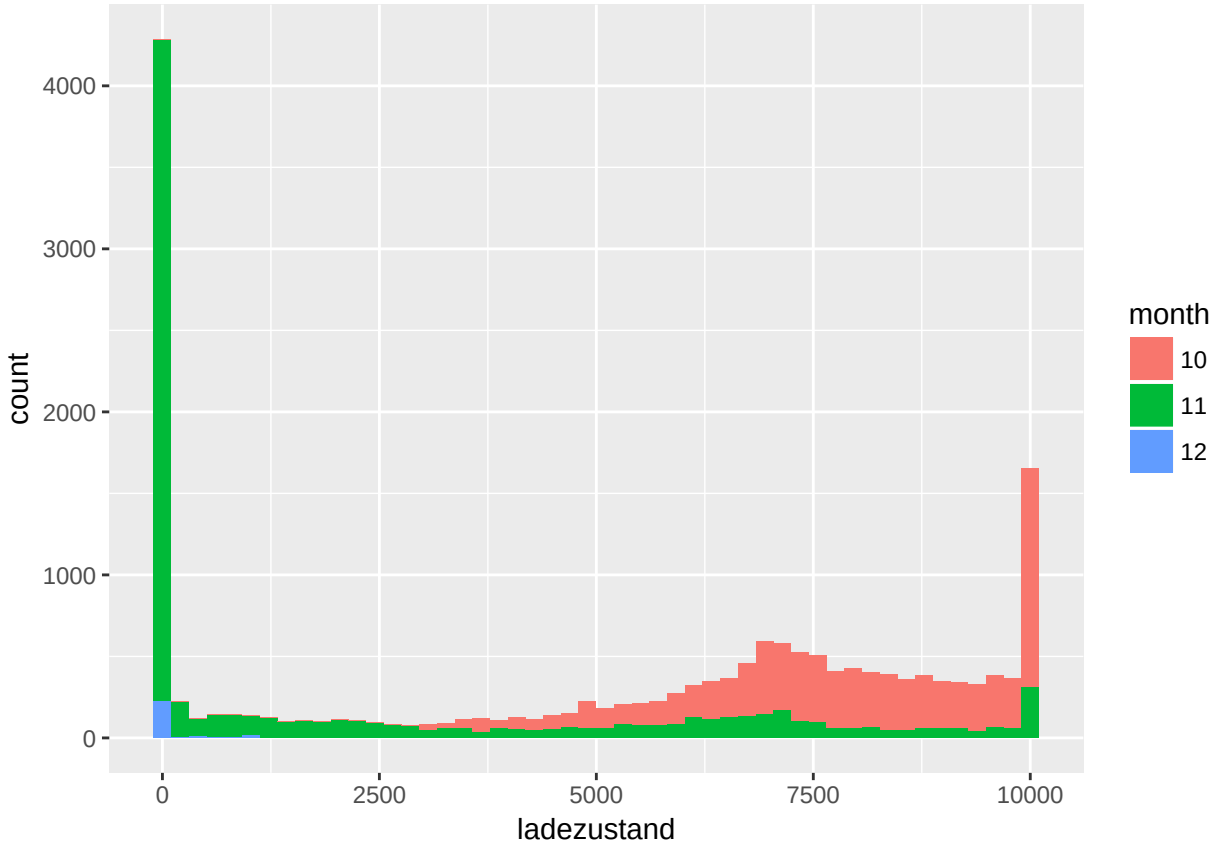



```
zusfg.long <- melt(zusfg , id = "month", measure = c("direktverbrauch", "batt_ladung", "netzeinspeisung

ggplot(zusfg.long, aes(month, value, fill = variable)) +
   geom_bar(stat = "identity", position = "stack") +
   labs(     x  = "Kalenderwoche",
             y  = "Energie (kWh)",
          fill  = " Art",
          title = "Erzeugung monatlich",
          subtitle = "differenziert nach Verwendung") +
   geom_text(aes(label=round(value,1)), data = zusfg.long, position = position_stack(vjust=0.5))
```



2.1.3 Batteriezustand

```
batt.zustand <- data %>%
  select(zeit, month, week, day,hour, ladezustand)
ggplot(batt.zustand) +
 #geom_bar(mapping = aes(x = ladezustand, fill=month ))
  stat_bin(mapping = aes(x = ladezustand, fill=month ), bins = 50)
```



2.2 Tägliche Minima und Maxima identifizieren - optional

```
# Minima und MAxima markieren frueher R_Min_Max_mark.R
 source("04_Auswertungen_Min_Max_tgl_Per.R")
#-------------------------------------------------------------------------------
```

```
# Minima und MAxima markieren frueher R_Min_Max_mark.R
 source("04_Auswertungen_Summ_Ent_Ladung_in_Tagesper.R")
```

```
## 04_Auswertungen_Summ_Ent_Ladung_in_Tagesper.R
```

2.3 Perioden zwischen horizontalen Niveaus bilden - Neutrale Zyklen

Ein *neutraler Zyklus* ist eine Lade-Entlade-Vorgang der von einem Ladezustand des Akkus ausgehend zu diesem zurückkehrt. Für diese ist es sinnvoll, Wirkungsgrade als Verhältnis von Output zu Input zu bilden.

"level" legt die Höhe des Ausgangszustands fest, der als Basis für die Berechnung von Wirkungsgraden dient. Ein solcher Zyklus kann jeweils über oder unter dem Ausgangslevel bleiben (später mit UP bzw. DOWN gekennzeichnet).

2.3.1 Bildung der Grundfunktionen

1. Initialisieren der Funktion "zyklus_daten_gen(xdata, l)" mit den Parametern xdata zur Übergabe der Daten und l zur Übergabe des Levels

```
Erzeugt die Spalten, die einen Zyklus mit einem Zähler charakteriseren und dessen Länge zählen:
zyklus --- len_of_zyklus
```

2. Initialisieren der Funktion “zyklus_summen_gen(xdata)” mit dem Parameter xdata zur Übergabe von data

```
Erzeugt die innerhalb eines Zyklus konstanten Werte:
max_level : max(ladezustand),
min_level : min(ladezustand),
hub_level : max_level - min_level,
mit_level : (min_level+max_level)/2,
durchsatz : hub_level/len_zyklus*12, Einheit  Wh zwischen Min und Max / Stunde
signum    : Wenn max_level über dem vorgegebenen Level  "UP" sonst "DOWN"
lev       : Der gewählte Level gespeichert in % im Hinblick auf die Verkettung der Daten zu mehrere
```

2.3.2 Zusammenfassung dieses Vorgangs

Dazu wird folgende Funktion definiert

```
zyklen_bilden <- function(xdata, x) {   # xdata =Datensatz, x Vorgabe eines Levels
  xdata = zyklus_daten_gen(xdata, x)
  xdata = zyklus_summen_gen(xdata,x)
  red_data = zyklus_reduzieren(xdata)
  return(red_data)
}
```

Sie gibt die Auswertung zurück mit jeweils einem Wert pro Zyklus.

2.4 Strecken monotoner Entladung

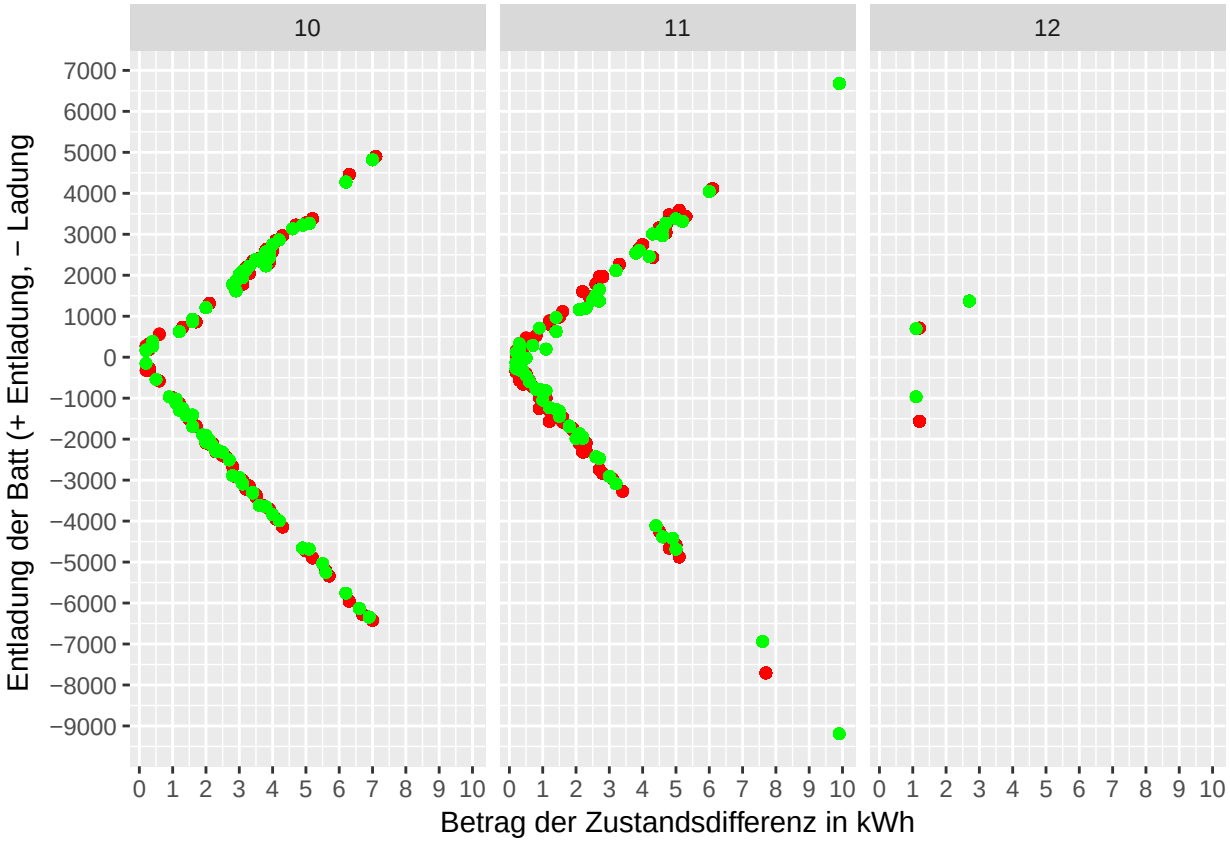
```
source("04_Auswertungen_Monotone_Entladung_finden.R")
```

```
temp <- monotonie_mark(data)
temp[is.na(temp)] <- 0

temp <- temp %>%
    mutate( r_mono  = mono,
            l_mono  = mono)

sp_vec <- temp %>%
    filter(mono != 0) %>%
    mutate(pos = mono*ct)      # pos enthält an Platz p den Wert +/- p wenn ein Anstieg/Abfall vorliegt

sp_vec   = sp_vec$pos           # Vektor der Sprünge
l_sp_vec = length(sp_vec)-1     # dessen Laenge -1

for (i in 1:l_sp_vec) {             # Auffüllen nach rechts  / links
    a <- sp_vec[i]                  # Sprung bei Pos a
    o <- sp_vec[i+1]                # nächster bei Betrag von o
    up <- abs(o)-1                  # letzte zu ändernde Position
    dn <- abs(a)+1                  # erste zu ändernde
    sa <- sign(a)
    so <- sign(o)
    for (j in dn:up ) {temp$l_mono[j] <-  so}
```

```
    for (j in dn:up ) {temp$r_mono[j] <-  sa}
}

temp    <- temp %>% select(-one_of(c("is_min","is_max","day_bat_in","day_bat_out","day_period_ladehub",'

temp <- temp %>%
   ungroup() %>%
   mutate(  rup   = ifelse(r_mono ==  1, 1, 0) ,
            rdown = ifelse(r_mono == -1, 1, 0),
            rsu   = ifelse(lag(rup)   != rup,   1, 0),
            rsd   = ifelse(lag(rdown) != rdown, 1, 0),
            rs    = rsu | rsd) %>%
   mutate(  lup   = ifelse(l_mono ==  1, 1, 0),
            ldown = ifelse(l_mono == -1, 1, 0),
            lsu   = ifelse(lag(lup)   != lup,   1, 0),
            lsd   = ifelse(lag(ldown) != ldown, 1, 0),
            ls    = lsu | lsd) %>%
   ungroup()

temp[is.na(temp)]<- 0

temp <- temp %>%
      mutate( lnr   = cumsum(ls),
              rnr   = cumsum(rs),
              eins = 1 ) %>%
      select(-one_of(c("lsu", "lsd", "ls","rsu", "rsd", "rs")))   %>%
      group_by(lnr) %>%
           mutate(l_input    = sum(batt_ladung),
                  l_output   = sum(batt_entladung),
                  l_netto_output = (l_output - l_input),
                  l_signum = sign(l_output-l_input),
                  l_entladung= max(ladezustand)-min(ladezustand),
                  l_dauer = sum(eins) /12 ) %>%  # Dauer in Stunden, vorher 1 entspricht 5 min
      ungroup() %>%
      group_by(rnr) %>%
           mutate(r_input    = sum(batt_ladung),
                  r_output   = sum(batt_entladung),
                  r_netto_output = (r_output - r_input),
                  r_signum = sign(r_output-r_input),
                  r_entladung= max(ladezustand)-min(ladezustand),
                  r_dauer = sum(eins) /12 ) %>%  # Dauer in Stunden, vorher 1 entspricht 5 min
      ungroup()

# workleft <- temp %>%
#       group_by(lnr) %>%
#       slice(1) %>%
#       ungroup()
#
# workright <- temp %>%
#       group_by (rnr) %>%
#       slice(1) %>%
#       ungroup()
#
```

```
# workleft %>%
#    filter(day >= "2017-10-01") %>%
#    filter(l_entladung < 10000 & l_entladung > 100) %>%
#    ggplot(aes(x = l_entladung, y = l_netto_output)) +
#    geom_point(aes(x = l_entladung, y = l_netto_output, color=l_dauer)) +
#    geom_smooth(mapping = aes(x = l_entladung, y = l_netto_output ),method=lm, se =FALSE) +
#    labs(
#       x = "Ladung / Entladung",
#       y = "netto_output",
#       color = "Dauer" ) +
#    facet_wrap(~ l_signum)
```

```
temp %>%
   filter(day >= "2017-10-01") %>%
   filter(r_entladung < 9990 & r_entladung > 100) %>%
   filter(l_entladung < 9990 & l_entladung > 100) %>%
   ggplot() +
   #ggplot(aes(x = r_entladung, y = r_netto_output)) +
   geom_point(aes(x = l_entladung/1000, y = l_netto_output), color= "red") +
   geom_point(aes(x = r_entladung/1000, y = r_netto_output), color= "green") +
   #geom_point(aes(x = r_entladung/1000, y = r_netto_output, color= as.logical((1+r_signum)/2))) +
   scale_y_continuous(breaks = seq(-10000, 10000, by = 1000)) +
   scale_x_continuous(breaks = seq(0, 10, by = 1)) +
   #geom_smooth(mapping = aes(x = l_entladung, y = l_netto_output )) + #,method=lm, se =FALSE) +
   #geom_smooth(mapping = aes(x = r_entladung, y = r_netto_output ),method=lm, se =FALSE) +
   labs(
     x = "Betrag der Zustandsdifferenz in kWh",
     y = "Entladung der Batt (+ Entladung, - Ladung ",
     color = "Vorzeichen" ) +
   facet_wrap(~ month)
```



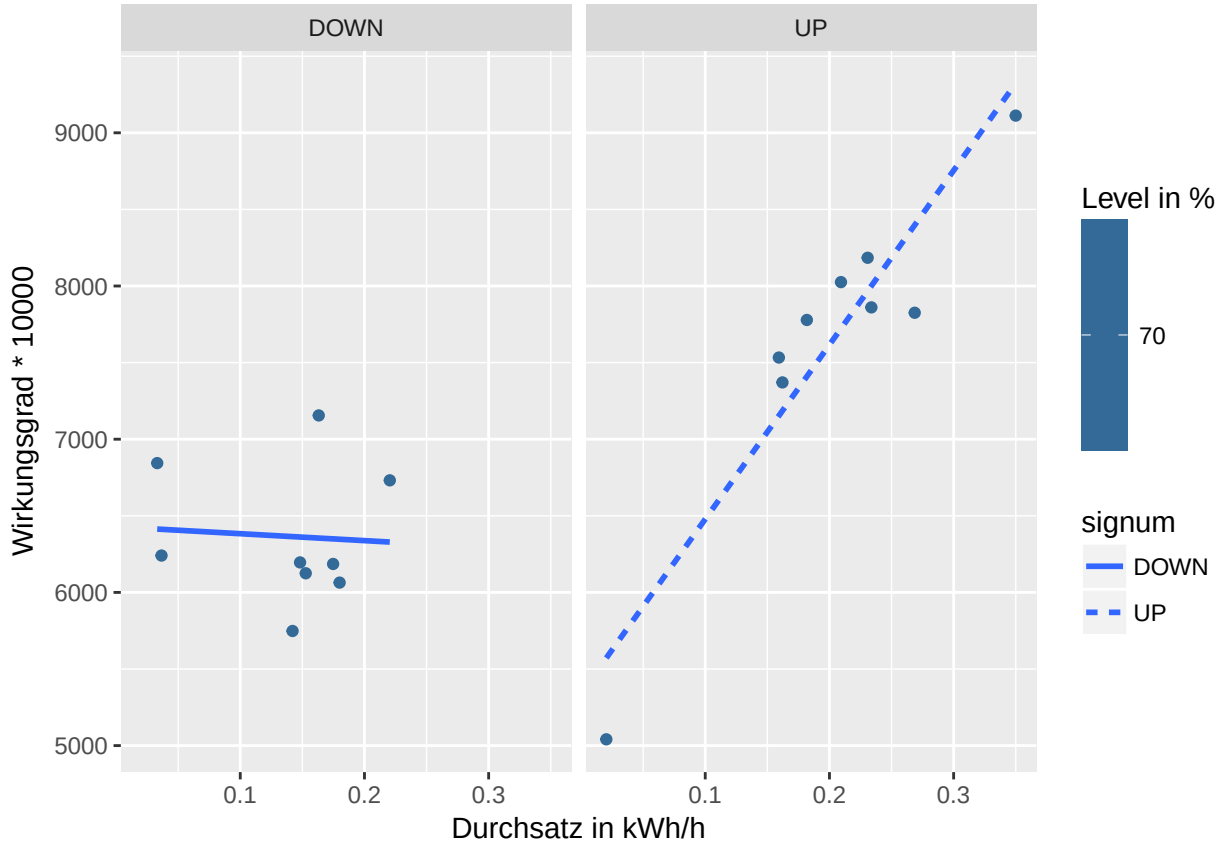
3 Graphische Auswertungen

3.1 Darstellung der Wirkungsgrade in Abhängigkeit von Durchsatz

Die maximale Energiedifferenz zwischen höchsten und niedrigsten Wert im Speicher in einer Halbperiode dividiert durch die Dauer der Halbperiode wird als (Energie-)Durchsatz bezeichnet angegeben in Wh/h. 'lev' bezeichnet dem gewählten level dividiert durch 1000.

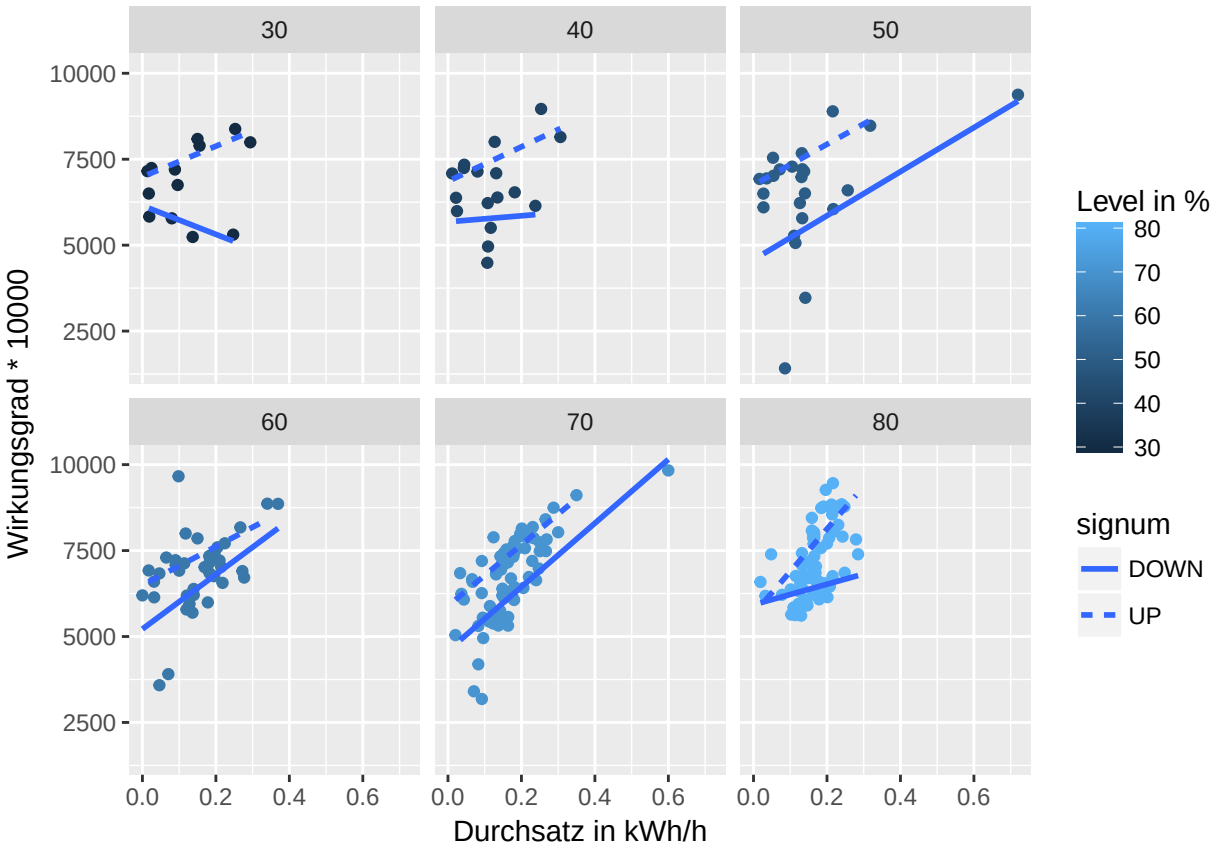
```
#-------------------------------------------------------
# Auswertung der Zyklen

proj_level = zyklen_bilden(data, 7000)

proj_level <- proj_level %>%
  filter(eta > 0 & eta <= 10000) %>%
  filter(day >= "2017-11-01")

proj_level %>%
   ggplot(aes(x = durchsatz/1000, y = eta)) +
   geom_point(aes(x = durchsatz/1000, y = eta, color=lev)) +
   geom_smooth(mapping = aes(x = durchsatz/1000, y = eta, linetype = signum ),method=lm, se =FALSE) +
   labs(
      x = "Durchsatz in kWh/h",
      y = "Wirkungsgrad * 10000",
```

```
      color = "Level in %" )+
  facet_wrap(~ signum)
```



Duch Wahl mehrerer Levelwerte erhält man folgende Darstellung

```
some_levels <- c(3000, 4000, 5000, 6000, 7000, 8000)
proj_level <- tibble()
for ( level in some_levels) {          # zu jedem level den Datensatz auswerten und reduzieren auf eine
  proj_level <- rbind(proj_level, zyklen_bilden(data, level)) # ueber mehrere Level aufsammeln
}
proj_level %>%
   filter(eta <= 10000 & eta  != 0) %>%
   filter(day >= "2017-10-01") %>%
   ggplot(aes(x = durchsatz/1000, y = eta)) +
   geom_point(aes(x = durchsatz/1000, y = eta, color=lev)) +
   geom_smooth(mapping = aes(x = durchsatz/1000, y = eta, linetype = signum ),method=lm, se =FALSE) +
   labs(
      x = "Durchsatz in kWh/h",
      y = "Wirkungsgrad * 10000",
      color = "Level in %"
   )  +                                        #+
   #geom_line(mapping = aes(x = durchsatz, y = eta, linetype = signum )) #+
   facet_wrap(~ lev)
```

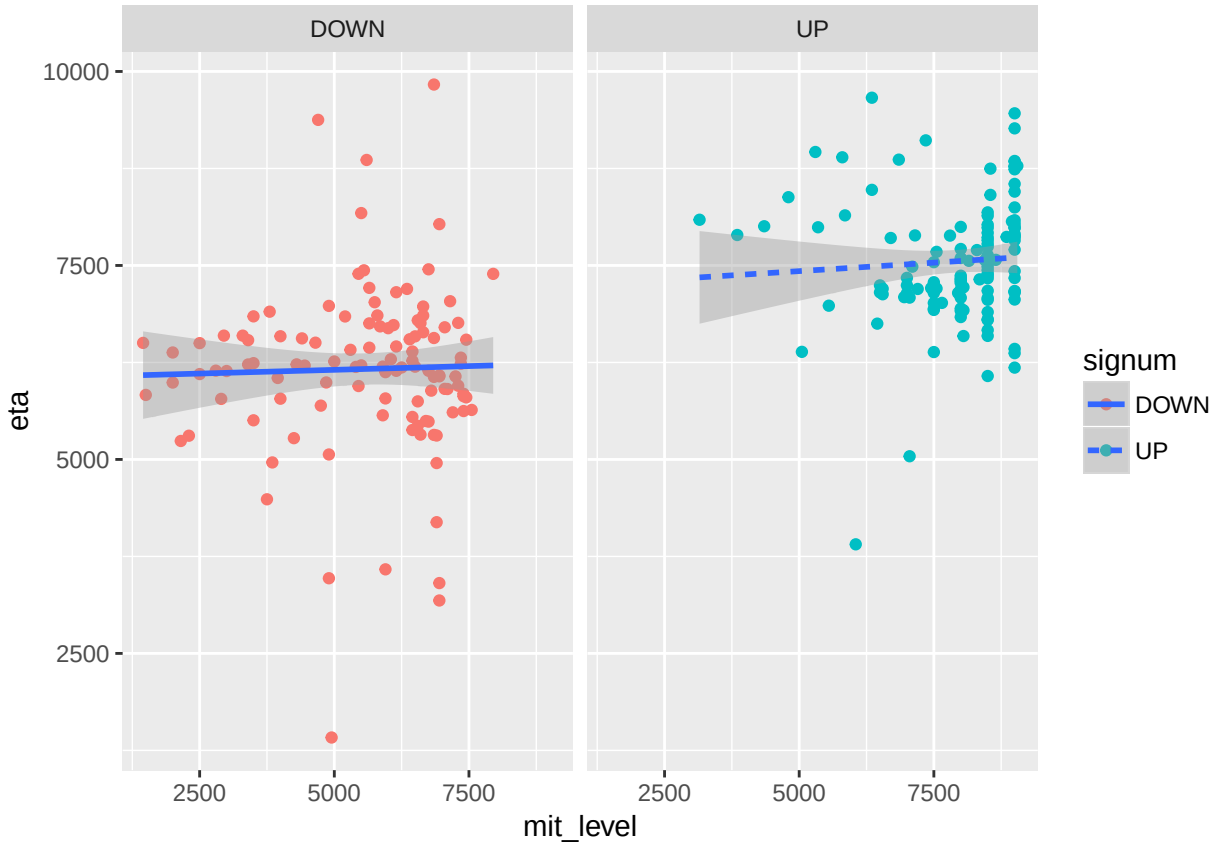


3.2 Darstellung der Wirkungsgrade in Abhängigkeit von der Mitte der Halbperiode

```
# ------------------------------------------------------------------ Proj_Level_Eta_vs_mit_level.R
source("05_Grafik_Eta_vs_Mitte.R", print.eval=TRUE)
```

```
## level  8000
## # A tibble: 109 x 29
##                   zeit batt_ladung batt_entladung ladezustand month  week
##                 <dttm>       <dbl>          <dbl>       <dbl> <chr> <dbl>
##  1 2017-10-01 00:40:00     0.00000     19.3333333           0    10    40
##  2 2017-10-01 14:10:00   115.08333      0.0000000        8000    10    40
##  3 2017-10-01 14:15:00    87.33333      0.1666667        8100    10    40
##  4 2017-10-02 00:15:00     0.00000      9.0833333        8000    10    40
##  5 2017-10-02 09:40:00   108.50000      0.0000000        8000    10    40
##  6 2017-10-02 09:45:00   163.50000      0.0000000        8200    10    40
##  7 2017-10-02 21:50:00     0.00000     23.5000000        8000    10    40
##  8 2017-10-03 14:30:00   172.08333      0.0000000        8000    10    40
##  9 2017-10-03 14:35:00   105.66667      0.0000000        8200    10    40
## 10 2017-10-03 21:45:00     0.00000     23.9166667        8000    10    40
## # ... with 99 more rows, and 23 more variables: day <date>, hour <dbl>,
## #   ladediff <dbl>, ct <dbl>, is_min <dbl>, is_max <dbl>, daypd <dbl>,
## #   len_daypd <dbl>, day_bat_in <dbl>, day_bat_out <dbl>,
## #   day_period_ladehub <dbl>, zyklus <dbl>, len_zyklus <dbl>,
```

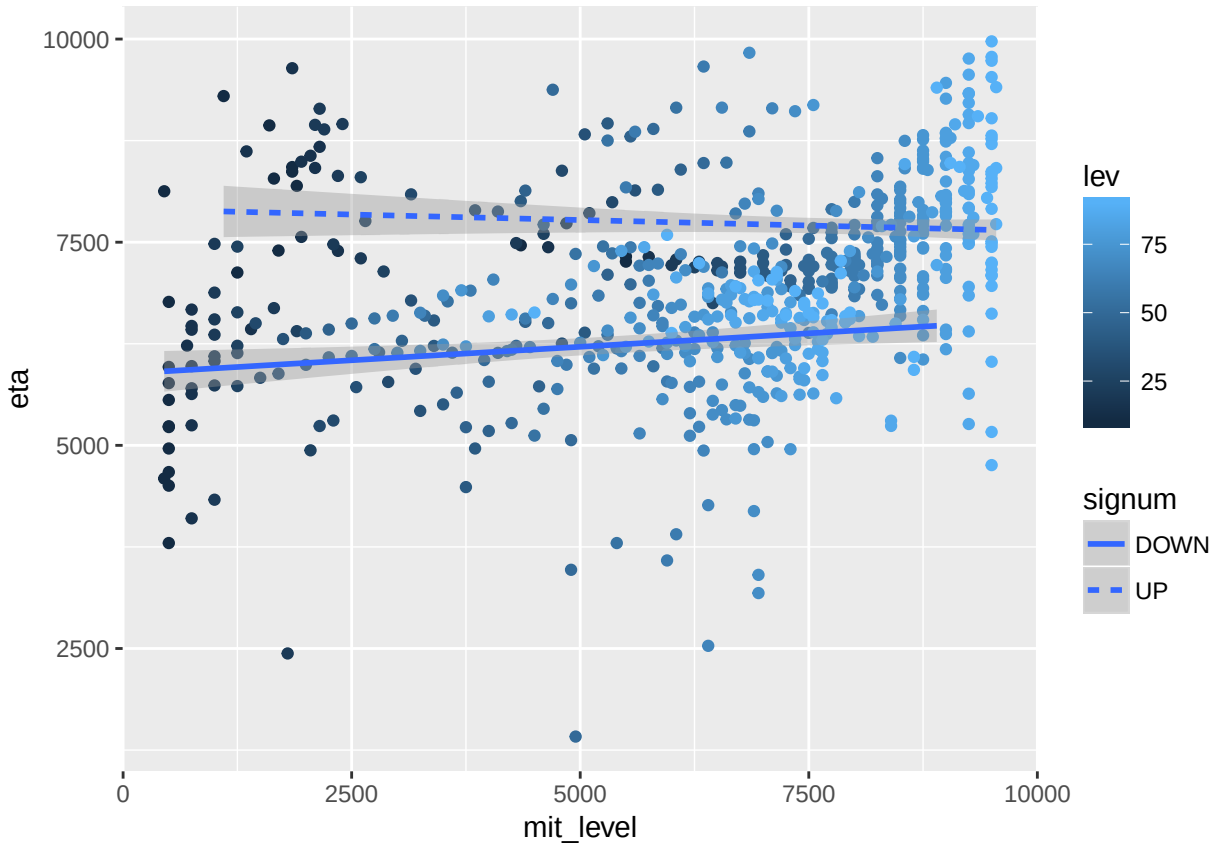
```
## #   lev_bat_in <dbl>, lev_bat_out <dbl>, eta <dbl>, max_level <dbl>,
## #   min_level <dbl>, hub_level <dbl>, mit_level <dbl>, durchsatz <dbl>,
## #   signum <chr>, lev <dbl>
```



3.2.1 Das Gleiche mit Aufsammeln von Daten zu mehreren Levels

Wirkungsgrade eta=0 oder eta > 10000 werden ausgeblendet.

```
some_levels <- c(1000, 1500, 2000, 2500, 3000, 3500, 4000, 4500, 5000,5500, 6000,6500, 7000,7500, 8000,
proj_level <- tibble()
for (level in some_levels) {
   proj_level <- rbind(proj_level, zyklen_bilden(data,level))
}
proj_level %>%
   filter(eta  != 0 & eta <= 10000) %>%
   ggplot(aes(x = mit_level, y = eta)) +
   geom_point(aes(x = mit_level, y = eta, color=lev)) +
   geom_smooth(mapping = aes(x = mit_level, y = eta, linetype = signum ), method=lm) #+
```




```
#geom_line(mapping = aes(x = durchsatz, y = eta, linetype = signum )) +
#facet_wrap(~ signum)
```