01 Basic R Syntax

Martin Hanewald

2019-02-19

Grundlegende Datentypen

```
num <- 5.1 # numeric
int <- 5L # integer
char <- '433' # character
char2 <- "slkjs" # character

logic <- TRUE # logical
logic2 <- TRUE | FALSE # logical
logic3 <- T #logical

date <- as.Date('2015-01-01')

null <-NULL
na <- NA
```

Auf Datentypen testen

```
null > 5
#> logical(0)
na > 5
#> [1] NA

is.numeric(num) # TRUE
#> [1] TRUE
is.integer(num) # FALSE
#> [1] FALSE

is.numeric(int) # TRUE
#> [1] TRUE
is.integer(int) # TRUE
#> [1] TRUE

is.character(char)
#> [1] TRUE

is.logical(F)
#> [1] TRUE
```

Datenkonstrukte

Vektoren

Die Funktion `c` (für combine) erzeugt Vektoren aus Einzelelementen oder anderen Vektoren. Dabei werden unterschiedliche Datentypen ineinander überführt (‘coercing’). Die Transformationsrichtung ist immer:

```
c(TRUE, TRUE, FALSE)
#> [1]  TRUE  TRUE FALSE
c(1,3,5,8)
#> [1] 1 3 5 8
c('a', 'b', 'c')
#> [1] "a" "b" "c"
c(num, char)
#> [1] "5.1" "433"
c(num, int)
#> [1] 5.1 5.0
c(logic, num)
#> [1] 1.0 5.1
c(logic, int)
#> [1] 1 5

vec <- c(num, int, logic, char)
vec
#> [1] "5.1"  "5"    "TRUE" "433"
```

Rechnen mit Vektoren

```
x <- 1:10
y <- 5:14

x + y
#>  [1]  6  8 10 12 14 16 18 20 22 24

x * y
#>  [1]   5  12  21  32  45  60  77  96 117 140

x / y
#>  [1] 0.2000000 0.3333333 0.4285714 0.5000000 0.5555556 0.6000000 0.6363636
#>  [8] 0.6666667 0.6923077 0.7142857
```

Recycling:

> Ein kürzere Vektor wird immer wiederholt wenn er mit einem längeren kombiniert wird.

```
y <- 1:2

x + y
#>  [1]  2  4  4  6  6  8  8 10 10 12

x * y
#>  [1]  1  4  3  8  5 12  7 16  9 20
```

Indizierung

```
x[5] # das fünfte Element
#> [1] 5

x[5:10] # das fünfte bis 10te Elemente
#> [1]  5  6  7  8  9 10
```

```r
x[x < 5] # alle Elemente deren Wert < 5
#> [1] 1 2 3 4

ind <- x < 4 | x > 9 # Kombination von logischen Operatoren
x[ind]
#> [1]  1  2  3 10

x[c(T, F, F)]
#> [1]  1  4  7 10

x[8:10] <- NA
```

Nützliche Funktionen zur Vektorerzeugung

```r
seq(5, 100, by = 5) # Sequenzen
#>  [1]   5  10  15  20  25  30  35  40  45  50  55  60  65  70  75  80  85
#> [18]  90  95 100
1:20 * 5 # äquivalent
#>  [1]   5  10  15  20  25  30  35  40  45  50  55  60  65  70  75  80  85
#> [18]  90  95 100

rep(1:5, 3) # Repeat 3 mal
#>  [1] 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5

seq_along(letters) # 1 bis Länge eines anderen Vektors
#>  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
#> [24] 24 25 26
```

Matrizen

```r
mat <- matrix(1:9, nrow=3) # numeric matrix
mat
#>      [,1] [,2] [,3]
#> [1,]    1    4    7
#> [2,]    2    5    8
#> [3,]    3    6    9

mat <- matrix(letters, nrow=2) # character matrix
mat
#>      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13]
#> [1,] "a"  "c"  "e"  "g"  "i"  "k"  "m"  "o"  "q"  "s"   "u"   "w"   "y"
#> [2,] "b"  "d"  "f"  "h"  "j"  "l"  "n"  "p"  "r"  "t"   "v"   "x"   "z"

# Indizierung von Matrizen: [Zeile, Spalte]
mat[2, 5:10] # 2. Zeile, 5.-10. Spalte
#> [1] "j" "l" "n" "p" "r" "t"
```

Listen

```r
l <- list(first = x, second = y, third = char) # 'Named list'
l
#> $first
#>  [1]  1  2  3  4  5  6  7 NA NA NA
#>
#> $second
```

```
#> [1] 1 2
#>
#> $third
#> [1] "433"

names(l)
#> [1] "first"  "second" "third"

l$first
#>  [1]  1  2  3  4  5  6  7 NA NA NA

l[1] # Das erste Listenelement (immer noch eine Liste)
#> $first
#>  [1]  1  2  3  4  5  6  7 NA NA NA

l[[1]][5] # Das fünfte Vektor Element des ersten Listenelements
#> [1] 5

a <- list()

a$nice <- function(x) sum(x)
a$other <- 5

a
#> $nice
#> function (x)
#> sum(x)
#>
#> $other
#> [1] 5
```

Dataframes

```
names <- c('Henri', 'Klaus', 'Tiffy')
age <- c(3, 21 , 55)
isadult <- age > 18

df <- data.frame(names, age, isadult)
df
#>   names age isadult
#> 1 Henri   3   FALSE
#> 2 Klaus  21    TRUE
#> 3 Tiffy  55    TRUE

df$age # Rückgabe ist Vektor
#> [1]  3 21 55

df['age'] # Rückgabe ist Data.Frame
#>   age
#> 1   3
#> 2  21
#> 3  55

df[['age']] # Rückgabe ist Vektor
#> [1]  3 21 55

df[,2] # Rückgabe ist Vektor
#> [1]  3 21 55

df[1,2] <- 22 # Änderung von Werten
```

```
df
#>   names age isadult
#> 1 Henri  22   FALSE
#> 2 Klaus  21    TRUE
#> 3 Tiffy  55    TRUE

str(df) # Datenstruktur
#> 'data.frame':    3 obs. of  3 variables:
#>  $ names  : Factor w/ 3 levels "Henri","Klaus",..: 1 2 3
#>  $ age    : num  22 21 55
#>  $ isadult: logi  FALSE TRUE TRUE

summary(df) # Summary Statistiken
#>    names        age         isadult
#>  Henri:1   Min.   :21.00   Mode :logical
#>  Klaus:1   1st Qu.:21.50   FALSE:1
#>  Tiffy:1   Median :22.00   TRUE :2
#>            Mean   :32.67
#>            3rd Qu.:38.50
#>            Max.   :55.00
```

02 Loops, If-then-else und Funktionen

Martin Hanewald

2019-02-19

IF THEN ELSE

```
dat <- 3

if(is.numeric(dat)){
    dat <- dat + 5
    cat('Die Zahl lautet', dat)
}else{
    cat(dat, 'ist keine Zahl!')
}
#> Die Zahl lautet 8
```

FOR LOOPS

```
names <- c('Harry', 'Bert', 'Murk')

for(n in names){
    cat(n, 'ist ein wunderschöner Name.\n')
}
#> Harry ist ein wunderschöner Name.
#> Bert ist ein wunderschöner Name.
#> Murk ist ein wunderschöner Name.

for(n in 1:length(names)){
    cat('Der', n, '. Name lautet', names[[n]], '\n')
}
#> Der 1 . Name lautet Harry
#> Der 2 . Name lautet Bert
#> Der 3 . Name lautet Murk

for(n in seq_along(names)){
    cat(n)
}
#> 123
```

FUNKTIONEN

```
myfunc <- function(){
    cat('Diese Funktion ist sinnlos.')
}

myfunc()
#> Diese Funktion ist sinnlos.

cylinder_volume <- function(r, h){
    vol <- pi * r^2 * h
    return(vol)
```

```
}

cylinder_volume(5.5, 20)
#> [1] 1900.664
```

APPLY

```
df <- data.frame(a=rnorm(10), b = rnorm(10), c = rnorm(10))

df
#>             a          b           c
#> 1  -1.4716206 -1.9918579  1.18819811
#> 2  -0.6865134 -0.5381026 -1.43548630
#> 3   0.2476627 -1.5368141  1.13263244
#> 4  -0.5301211 -1.4121818 -0.04986148
#> 5   0.3738892  0.4117586  0.71504782
#> 6  -1.2518694  0.1525651 -0.73350210
#> 7  -1.3199694 -0.4902913 -1.18916517
#> 8   0.4559150  1.9933542 -0.53554275
#> 9   0.4909932  1.3147484 -0.13624744
#> 10 -0.1042614  0.5873565  0.23527954
```

Herausforderung: Berechne Mittelwert über

a. alle Zeilen
b. alle Spalten

Mögliche Lösung: Einzelaufruf

```
mean(df[,1])
#> [1] -0.3795895
mean(df[,2])
#> [1] -0.1509465
mean(df[,3])
#> [1] -0.08086473

# Für Berechnung der Zeilen muss erst noch transponiert werden.
mean(t(df[1,]))
#> [1] -0.7584268
mean(t(df[2,]))
#> [1] -0.8867008
mean(t(df[3,]))
#> [1] -0.05217299
```

Oder: Schleifenkonstruktion

```
for(k in 1:3){
    cat(mean(df[,k]))
    cat(mean(t(df[k,])))
}
#> -0.3795895-0.7584268-0.1509465-0.8867008-0.08086473-0.05217299
```

Aber es geht auch eleganter:

```
apply(df, 1, mean)
#>  [1] -0.75842679 -0.88670075 -0.05217299 -0.66405481  0.50023187
#>  [6] -0.61093546 -0.99980863  0.63790883  0.55649805  0.23945824
apply(df, 2, mean)
```

```
#>           a           b           c
#> -0.37958952 -0.15094648 -0.08086473
```

Die `apply` Funktion wendet die Funktion `mean` über alle Zeilen (`1`) oder Spalten (`2`) von `df` an.

Aufgabe: Schreibt eine Funktion die die Fibonacci Reihe erzeugt (20 min)

Fibonacci: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55

```
fibonacci <- function(x){
    ans <- rep(1,x)
    if(x > 2){
        for(k in 3:x){
            ans[k] <- ans[k-2] + ans[k-1]
        }
    }
    return(c(0,ans))
}

fibonacci(10)
#>  [1]  0  1  1  2  3  5  8 13 21 34 55
```

03 Datenaufbereitung mit tidyR

Martin Hanewald

2019-02-19

Packages

```
library(tidyverse)
library(knitr)
```

Überblick

Die gesamte Paketbibliothek `tidyverse` basiert auf dem Konzept von `tidy data`, (im Gegensatz zu `messy data`).

Ein Datensatz ist `tidy`, wenn

- Jede Variable eine eigene Spalte hat
- Jede Messung einer Variable in einer eigenen Zeile zu finden ist
- Mehrere Tabellen über eine eindeutige ID verknüpft werden können.

Die wichtigsten Funktionen von `tidyr`:

- `gather()`: Spaltenüberschriften zu Variablen
- `spread()`: Umkehrfunktion zu `gather`
- `separate()`: Textspalten auftrennen
- `unite()`: Umkehrfunktion zu `unite`

Beispiele von messy-Data

Variable “Year” ist in Spaltenüberschriften

```
data("table4a")
table4a %>% kable()
```

country	1999	2000
Afghanistan	745	2666
Brazil	37737	80488
China	212258	213766

Anwendung von `gather`:

```
table4a %>%
    gather(year, count, -country) %>%
    kable()
```

country	year	count
Afghanistan	1999	745
Brazil	1999	37737
China	1999	212258

country	year	count
Afghanistan	2000	2666
Brazil	2000	80488
China	2000	213766

In Spalte `rate` sind mehrere Variablen enthalten und die Variable `year` ist auf zwei Spalten verteilt.

```
data("table5")
table5 %>% kable()
```

country	century	year	rate
Afghanistan	19	99	745/19987071
Afghanistan	20	00	2666/20595360
Brazil	19	99	37737/172006362
Brazil	20	00	80488/174504898
China	19	99	212258/1272915272
China	20	00	213766/1280428583

Anwendung von `separate` und `unite`

```
table5 %>%
    separate(rate, into = c("cases", "population")) %>%
    unite(year, century, year, sep="") %>%
    kable()
```

country	year	cases	population
Afghanistan	1999	745	19987071
Afghanistan	2000	2666	20595360
Brazil	1999	37737	172006362
Brazil	2000	80488	174504898
China	1999	212258	1272915272
China	2000	213766	1280428583

Aufgabe: Transformiere den WHO Datensatz in das tidy Format

```
who_messy <- read_csv2('who_messy.csv')
```

Show 10 entries Search:

	country	iso2	iso3	year	ep_f_014	ep_f_1524	ep_f_2534	ep_f_35
1	Luxembourg	LU	LUX	1997				
2	Marshall Islands	MH	MHL	2005				

3	Micronesia (Federated States of)	FM	FSM	2007	1	0	2	
4	Yemen	YE	YEM	1997				
5	Cook Islands	CK	COK	2000				
6	Tunisia	TN	TUN	2005				
7	Angola	AO	AGO	2003				
8	Cook Islands	CK	COK	2003				
9	Ghana	GH	GHA	2002				
10	Seychelles	SC	SYC	2009	0	0	0	

Showing 1 to 10 of 100 entries Previous 1 2 3 4 5 … 10 Next

Aus Dokumentation:

> The data uses the original codes given by the World Health Organization. The column names for columns five through 60 are made by combining a code for method of diagnosis (rel = relapse, sn = negative pulmonary smear, sp = positive pulmonary smear, ep = extrapulmonary) to a code for gender (f = female, m = male) to a code for age group (014 = 0-14 yrs of age, 1524 = 15-24 years of age, 2534 = 25 to 34 years of age, 3544 = 35 to 44 years of age, 4554 = 45 to 54 years of age, 5564 = 55 to 64 years of age, 65 = 65 years of age or older).

Lösung

```
who_tidy <- who_messy %>%
    gather(col, value, -(country:year)) %>%
    separate(col, into = c('method', 'sex', 'age')) %>%
    drop_na()
```

Show 10 entries Search:

	country	iso2	iso3	year	method	sex	age	value
1	Hungary	HU	HUN	2010	sp	f	014	0
2	Argentina	AR	ARG	2012	ep	m	1524	128
3	Solomon Islands	SB	SLB	2009	sn	m	4554	6
4	Germany	DE	DEU	2001	sp	m	1524	3
5	El Salvador	SV	SLV	2012	sn	m	2534	28
6	Saint Vincent and the Grenadines	VC	VCT	2005	sp	f	5564	0
7	China, Hong Kong SAR	HK	HKG	2005	sn	f	014	14
8	Canada	CA	CAN	2000	sp	f	65	66

9	Nauru	NR	NRU	2003	sp		m	014	0
10	Lebanon	LB	LBN	1996	sp		m	1524	28

Showing 1 to 10 of 100 entries Previous 1 2 3 4 5 … 10 Next

```
```

04 Datenmanipulation mit dplyR

Martin Hanewald

2019-02-19

Packages

```
library(tidyverse)
library(nycflights13)
library(rvest)
library(knitr)
```

Überblick

Basis Operationen von `dplyr`:

- `filter()` zum Filtern nach Werten
- `arrange()` zum Sortieren
- `select()` und `rename()` zum Auswählen von Spalten und Umbenennen
- `mutate()` und `transmute()` zum Erzeugen neuer Spalten
- `group_by()` zum Definieren einer Gruppierungsebene
- `summarise()`zum Aggregieren von Kennzahlen auf einer Gruppierungsebene

Demonstration mit Star Wars Charakteren:

Welche Charaktere spielen am häufigsten mit?

```
data(starwars)

ans <- starwars %>% select(name, films) %>%
    unnest() %>%
    group_by(name) %>%
    count() %>% arrange(desc(n)) %>%
    head(10) %>%
    kable()
```

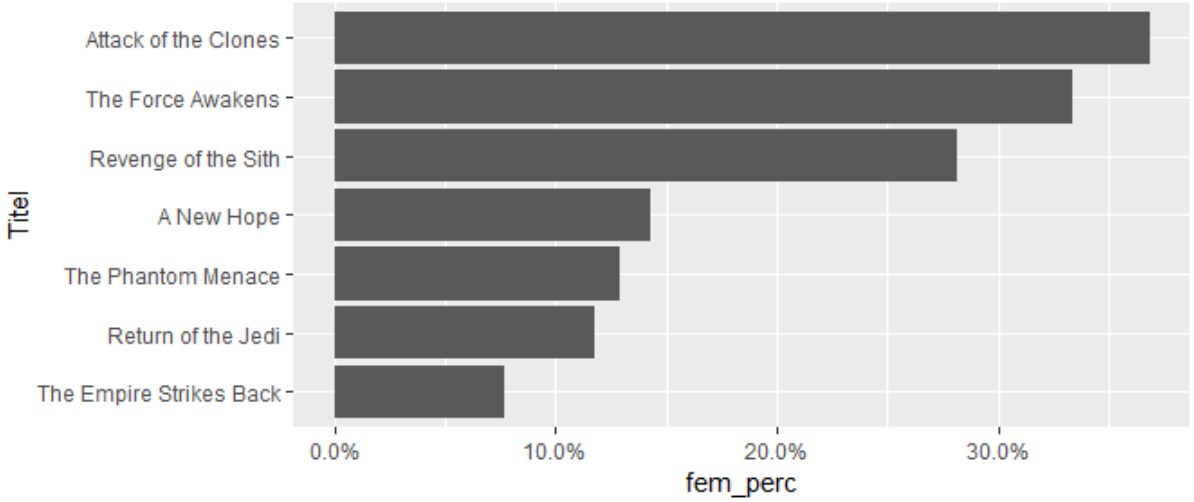
Berechne die Frauenquote pro Film ?

```
ans <- starwars %>% select(name, gender, films) %>% unnest %>%
    group_by(films, gender) %>% count() %>%
    filter(gender %in% c('female', 'male')) %>%
    spread(gender, n) %>%
    mutate(fem_perc = female / (female + male)) %>%
    select(films, fem_perc) %>%
    arrange(desc(fem_perc))

ans %>% kable()
```

films	fem_perc

films	fem_perc
Attack of the Clones	0.3684211
The Force Awakens	0.3333333
Revenge of the Sith	0.2812500
A New Hope	0.1428571
The Phantom Menace	0.1290323
Return of the Jedi	0.1176471
The Empire Strikes Back	0.0769231

```
ans %>%
    ggplot(aes(films %>% fct_reorder(fem_perc), fem_perc)) +
        geom_col() + coord_flip() + scale_y_continuous(labels=scales::percent)+
        labs(x='Titel', 'Anteil Frauen')
```



Aufgaben: Analysiere Flugdaten

Datensatz `flights` aus Package `nycflights13`.

```
data(flights)
```

Show 10 entries Search:

	year	month	day	dep_time	sched_dep_time	dep_delay	arr_time
1	2013	10	6	1454	1459	-5	1720
2	2013	5	11	1422	1426	-4	1723
3	2013	6	4	801	806	-5	1000
4	2013	1	4	1540	1545	-5	1850

5	2013	7	10	1311	1315	-4	1446
6	2013	11	2	911	915	-4	1114
7	2013	8	16	555	600	-5	710
8	2013	9	15	1002	1000	2	1442
9	2013	4	26	1346	1259	47	1628
10	2013	2	25	1853	1900	-7	1957

Showing 1 to 10 of 10 entries

Previous 1 Next

Welches sind die beliebtesten Reiseziele?

Liste die Top 10

```
flights %>%
    group_by(dest) %>%
    count() %>%
    arrange(desc(n)) %>%
    head(10) %>%
    kable()
```

dest	n
ORD	17283
ATL	17215
LAX	16174
BOS	15508
MCO	14082
CLT	14064
SFO	13331
FLL	12055
MIA	11728
DCA	9705

Welches sind die unpünktlichsten Fluggesellschaften

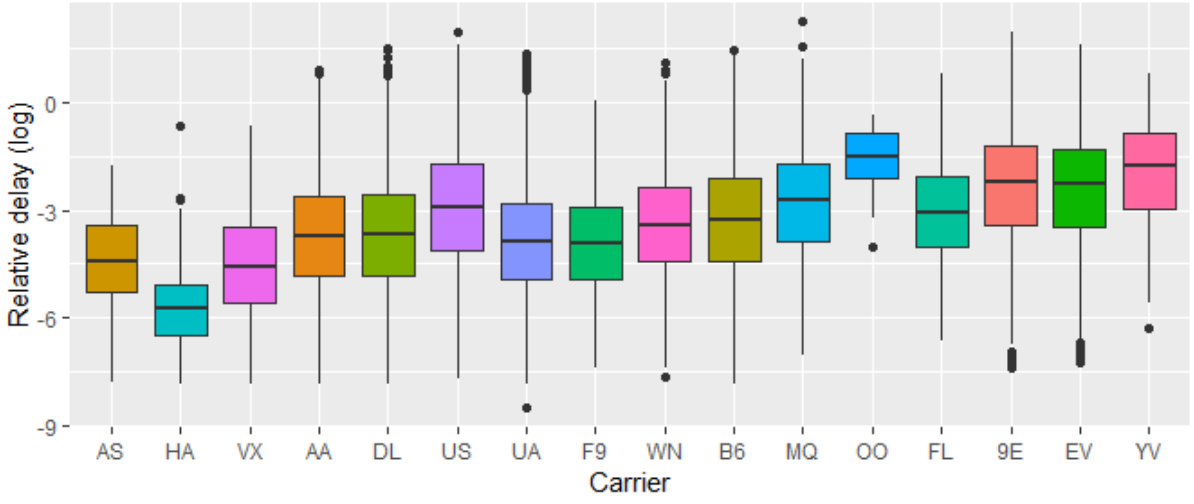
- Berechne die mittlere Verspätung im Verhältnis zur Strecke
- Sortiere nach schlechtesten Fluggesellschaften

```
ans <- flights %>%
    mutate(tot_delay = dep_delay + arr_delay) %>%
    mutate(rel_delay = tot_delay / distance) %>%
    group_by(carrier) %>%
```

```
    summarise(rel_delay = mean(rel_delay, na.rm =T),
              count = n()) %>%
    arrange(desc(rel_delay))


ans %>% head(10) %>% kable()
```

carrier	rel_delay	count
YV	0.1125663	601
EV	0.0854153	54173
9E	0.0697272	18460
FL	0.0637883	3260
OO	0.0498792	32
MQ	0.0442328	26397
B6	0.0383656	54635
WN	0.0317897	12275
F9	0.0260012	685
UA	0.0172654	58665

```
# Darstellung als Boxplot
flights %>%
    mutate(tot_delay = dep_delay + arr_delay) %>%
    mutate(rel_delay = tot_delay / distance) %>%
    ggplot(aes(carrier %>% fct_reorder(rel_delay, mean, na.rm=T), log(rel_delay), fill=carrier)) +
    geom_boxplot() + labs(y='Relative delay (log)', x='Carrier') +
    theme(legend.position="none")
#> Warning in log(rel_delay): NaNs wurden erzeugt

#> Warning in log(rel_delay): NaNs wurden erzeugt
#> Warning: Removed 201717 rows containing non-finite values (stat_boxplot).
```



Bonus: Auflösung des IATA Codes in Namen

```
url <- "https://aspmhelp.faa.gov/index.php/ASQP_:_Carrier_Codes_And_Names"
carrier_codes <- url %>%
    read_html() %>%
    html_table() %>%
    .[[1]]

ans %>%
    left_join(carrier_codes, by=c('carrier'='IATA Carrier Code')) %>%
    kable()
```

carrier	rel_delay	count	ICAO Carrier Code	Carrier Name
YV	0.1125663	601	ASH	Mesa Airlines
EV	0.0854153	54173	CAA	Atlantic Southeast Airlines
9E	0.0697272	18460	FLG	Pinnacle Airlines
FL	0.0637883	3260	TRS	AirTran Airways
OO	0.0498792	32	SKW	SkyWest Airlines
MQ	0.0442328	26397	EGF	American Eagle
B6	0.0383656	54635	JBU	JetBlue Airways
WN	0.0317897	12275	SWA	Southwest Airlines
F9	0.0260012	685	FFT	Frontier
UA	0.0172654	58665	UAL	United Airlines
US	0.0168932	20536	USA	US Airways
DL	0.0132526	48110	DAL	Delta Air Lines
AA	0.0087874	32729	AAL	American Airlines
VX	0.0057073	5162	NA	NA
HA	-0.0004043	342	HAL	Hawaiian Airlines
AS	-0.0017070	714	ASA	Alaska Airlines

05 Datumsformatierung mit lubridate

Martin Hanewald

2019-02-19

Packages

```
library(tidyverse)
library(lubridate)
library(knitr)
```

Syntax

Ein übliches Datum in Textformat

```
d <- c('31. Juli 2017 15:00')
d
#> [1] "31. Juli 2017 15:00"
```

Konvertierung in `datetime`

```
dttm <- dmy_hm(d)
dttm
#> [1] "2017-07-31 15:00:00 UTC"
```

Konvertierung in `date`

```
date <- dmy_hm(d) %>% as_date()
date
#> [1] "2017-07-31"
```

Repräsentation als numerischer Wert

```
dttm %>% as.numeric()
#> [1] 1501513200
date %>% as.numeric()
#> [1] 17378
```

Konvertierung aus einer Quartalsangabe

```
q <- yq('2017-Q3')
q
#> [1] "2017-07-01"
```

Nur eine Zeitangabe

```
hms('10:00:00')
#> [1] "10H 0M 0S"
```

Erzeugung einer Datums/Zeitsequenz

```
seq(ymd_h('2018-01-01 00'), ymd_h('2018-01-03 00'), by = '1 hours')
#>  [1] "2018-01-01 00:00:00 UTC" "2018-01-01 01:00:00 UTC"
#>  [3] "2018-01-01 02:00:00 UTC" "2018-01-01 03:00:00 UTC"
#>  [5] "2018-01-01 04:00:00 UTC" "2018-01-01 05:00:00 UTC"
#>  [7] "2018-01-01 06:00:00 UTC" "2018-01-01 07:00:00 UTC"
#>  [9] "2018-01-01 08:00:00 UTC" "2018-01-01 09:00:00 UTC"
#> [11] "2018-01-01 10:00:00 UTC" "2018-01-01 11:00:00 UTC"
#> [13] "2018-01-01 12:00:00 UTC" "2018-01-01 13:00:00 UTC"
#> [15] "2018-01-01 14:00:00 UTC" "2018-01-01 15:00:00 UTC"
#> [17] "2018-01-01 16:00:00 UTC" "2018-01-01 17:00:00 UTC"
#> [19] "2018-01-01 18:00:00 UTC" "2018-01-01 19:00:00 UTC"
#> [21] "2018-01-01 20:00:00 UTC" "2018-01-01 21:00:00 UTC"
#> [23] "2018-01-01 22:00:00 UTC" "2018-01-01 23:00:00 UTC"
#> [25] "2018-01-02 00:00:00 UTC" "2018-01-02 01:00:00 UTC"
#> [27] "2018-01-02 02:00:00 UTC" "2018-01-02 03:00:00 UTC"
#> [29] "2018-01-02 04:00:00 UTC" "2018-01-02 05:00:00 UTC"
#> [31] "2018-01-02 06:00:00 UTC" "2018-01-02 07:00:00 UTC"
#> [33] "2018-01-02 08:00:00 UTC" "2018-01-02 09:00:00 UTC"
#> [35] "2018-01-02 10:00:00 UTC" "2018-01-02 11:00:00 UTC"
#> [37] "2018-01-02 12:00:00 UTC" "2018-01-02 13:00:00 UTC"
#> [39] "2018-01-02 14:00:00 UTC" "2018-01-02 15:00:00 UTC"
#> [41] "2018-01-02 16:00:00 UTC" "2018-01-02 17:00:00 UTC"
#> [43] "2018-01-02 18:00:00 UTC" "2018-01-02 19:00:00 UTC"
#> [45] "2018-01-02 20:00:00 UTC" "2018-01-02 21:00:00 UTC"
#> [47] "2018-01-02 22:00:00 UTC" "2018-01-02 23:00:00 UTC"
#> [49] "2018-01-03 00:00:00 UTC"
```

Wochentage

```
wday(date, label=T)
#> [1] Mo
#> Levels: So < Mo < Di < Mi < Do < Fr < Sa
```

Eine Zeichenkette mit dem Format `YYYY-MM` erzeugen.

```
glue::glue('{year(date)}-{str_pad(month(date),2, pad=0)}')
#> 2017-07
date %>% format('%Y-%m')
#> [1] "2017-07"
```

Aufgabe: Erzeuge ein tibble mit folgendem Layout

Date	Quarter	Month	Month_label	Season
2019-01-01	Q1	2019-01	Jan	Winter
2019-02-01	Q1	2019-02	Feb	Winter
2019-03-01	Q1	2019-03	Mrz	Frühling
2019-04-01	Q2	2019-04	Apr	Frühling
2019-05-01	Q2	2019-05	Mai	Frühling
2019-06-01	Q2	2019-06	Jun	Sommer
2019-07-01	Q3	2019-07	Jul	Sommer
2019-08-01	Q3	2019-08	Aug	Sommer
2019-09-01	Q3	2019-09	Sep	Herbst
2019-10-01	Q4	2019-10	Okt	Herbst

Date	Quarter	Month	Month_label	Season
2019-11-01	Q4	2019-11	Nov	Herbst
2019-12-01	Q4	2019-12	Dez	Winter

Lösung:

```
season <- function(x){
    if(!is.Date(x)) return(NA)
    m <- month(x)
    if(m %in% c(12,1,2)) return('Winter')
    if(m %in% 3:5) return('Frühling')
    if(m %in% 6:8) return('Sommer')
    if(m %in% 9:11) return('Herbst')
}

tibble(Date = seq(ymd('2019-01-01'), ymd('2019-12-01'), by = '1 month')) %>%
    mutate(Quarter = paste0('Q',quarter(Date))) %>%
    mutate(Month = Date %>% format('%Y-%m')) %>%
    mutate(Month_label = Date %>% month(label=T)) %>%
    mutate(Season = sapply(Date, season))
```

06 Textmanipulation mit stringR

Martin Hanewald

2019-02-19

Packages

```
library(tidyverse)
library(knitr)
```

Aufgabe: Zähle alle Vorkommnisse von *fruit* in *sentences*

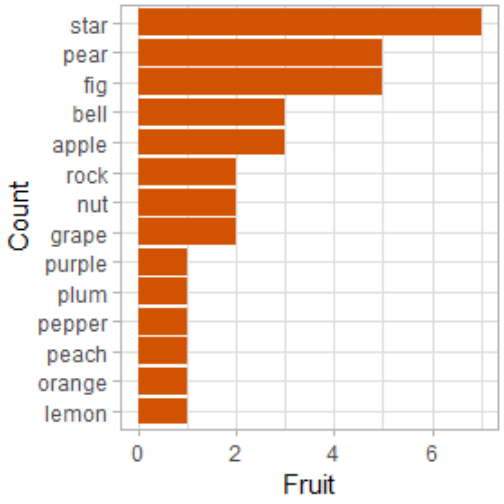
```
data(fruit)
head(fruit)
#> [1] "apple"       "apricot"     "avocado"     "banana"      "bell pepper"
#> [6] "bilberry"
data(sentences)
head(sentences)
#> [1] "The birch canoe slid on the smooth planks."
#> [2] "Glue the sheet to the dark blue background."
#> [3] "It's easy to tell the depth of a well."
#> [4] "These days a chicken leg is a rare dish."
#> [5] "Rice is often served in round bowls."
#> [6] "The juice of lemons makes fine punch."
```

Ziel

fruit	count
star	7
fig	5
pear	5
apple	3
bell	3
grape	2
nut	2
rock	2
pepper	1
orange	1
lemon	1
peach	1
plum	1
purple	1

Lösung

```
words <- fruit %>% str_split(" ") %>% unlist()

ans <- list()

for(w in words){
    ans[[w]] <- str_detect(tolower(sentences), tolower(w)) %>% sum()
}

result <- ans %>% as_tibble() %>%
    gather(fruit, ct) %>%
    filter(ct > 0) %>%
    arrange(desc(ct))

result %>%
    ggplot(aes(fruit %>% fct_reorder(ct), ct)) + geom_col(fill='#d15200') + coord_flip() +
    theme_light() + labs(x = 'Count', y = 'Fruit')
```



Alternative mit *apply*

```
### without for loop

fruit %>%
    str_split(" ") %>%
    unlist() %>%
    unique() %>%
    sapply(function(x) str_detect(tolower(sentences), x) %>% sum(), simplify = F) %>%
    as_tibble() %>%
    gather(fruit, count) %>%
    filter(count > 0) %>%
    arrange(desc(count))
#> # A tibble: 14 x 2
#>    fruit  count
#>    <chr>  <int>
#>  1 star       7
#>  2 fig        5
#>  3 pear       5
#>  4 apple      3
#>  5 bell       3
#>  6 grape      2
#>  7 nut        2
```

```
#>  8 rock       2
#>  9 pepper     1
#> 10 orange     1
#> 11 lemon      1
#> 12 peach      1
#> 13 plum       1
#> 14 purple     1
```

07 Erstellung von Grafiken mit ggplot2

Martin Hanewald

2019-02-19

Packages

```
library(tidyverse)
library(knitr)
library(DT)
```

Überblick

Das Paket `ggplot2` ist die meistgenutze Grafikbibliothek in R. Sein modularer Aufbau in `aesthetics`, `coordinates` und `geometries` erlaubt beliebige Freiheit in der Gestaltung von Plots.

Dataset

Datensatz `midwest` aus dem Package `ggplot2` enthält Daten einer Volkszählung.

```
data(midwest)
# Umwandlung einiger Variablen in Datentyp 'factor'
midwest <- midwest %>% mutate_at(vars(county, state, inmetro, category), as.factor)
# Show sample
sample_n(midwest, 10) %>% DT::datatable(width = 700, options=list(scrollX = TRUE))
```

Show 10 entries Search:

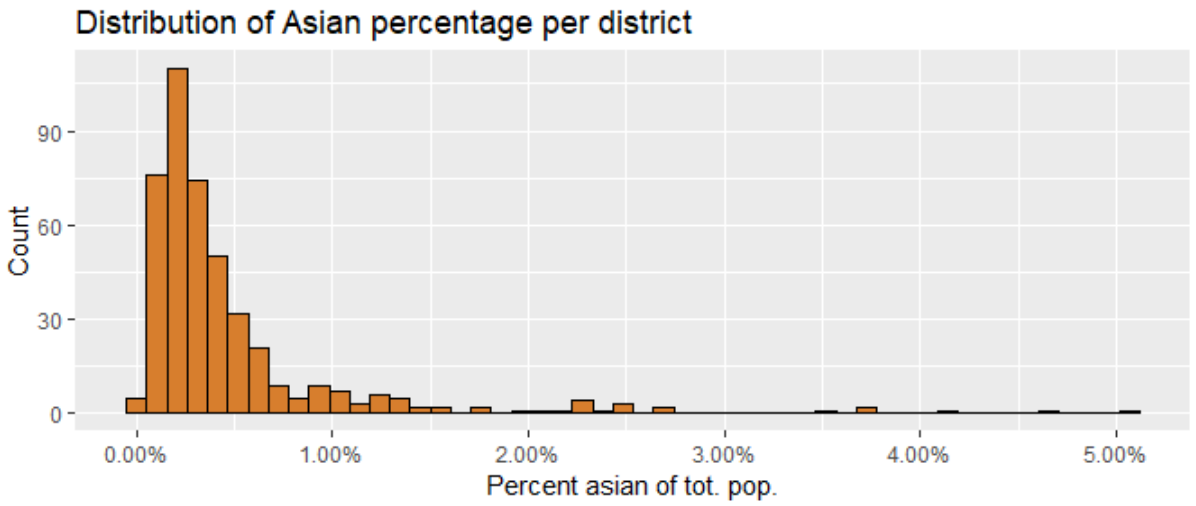
	PID	county	state	area	poptotal	popdensity	popwhite	popbla
1	691	HAMILTON	IN	0.024	108936	4539	106764	
2	705	KOSCIUSKO	IN	0.032	65294	2040.4375	64058	
3	2014	AUGLAIZE	OH	0.024	44585	1857.70833	44225	
4	1278	WAYNE	MI	0.035	2111687	60333.9143	1212007	849
5	742	TIPTON	IN	0.016	16119	1007.4375	15990	
6	1236	KALKASKA	MI	0.033	13497	409	13321	
7	3003	GREEN	WI	0.034	30339	892.323529	30173	
8	670	CARROLL	IN	0.022	18809	854.954545	18720	
9	2995	DOOR	WI	0.028	25690	917.5	25387	
10	2095	WOOD	OH	0.037	113269	3061.32432	109303	1

Showing 1 to 10 of 10 entries Previous 1 Next

Histogram

Verteilung einer numerischen Variable

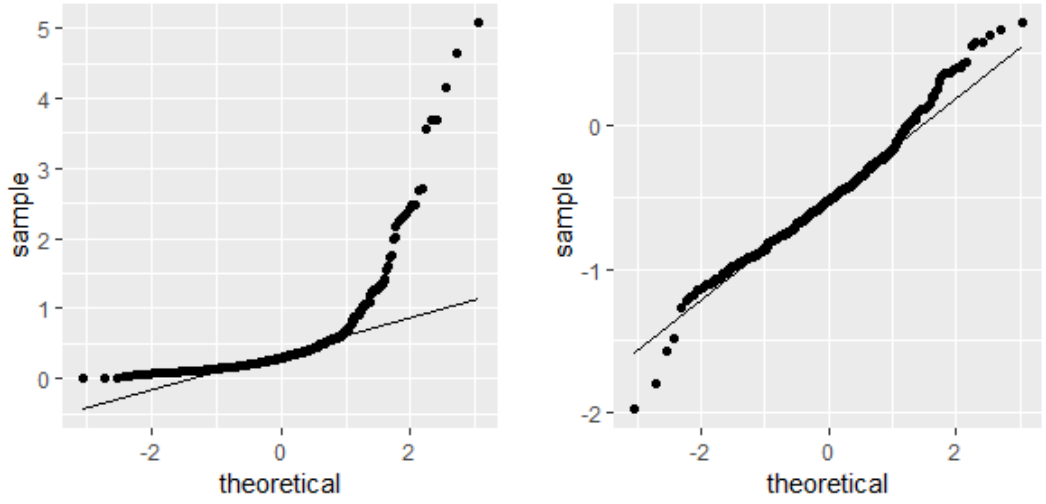
```
midwest %>%
    ggplot(aes(x = percasian / 100)) +
    geom_histogram(bins = 50, color=1, fill='#d77e2d') +
    scale_x_continuous(labels=scales::percent) +
    labs(x='Percent asian of tot. pop.', y = 'Count',
        title='Distribution of Asian percentage per district')
```



Quantilsplot

Vergleich einer Verteilung mit Normalverteilung

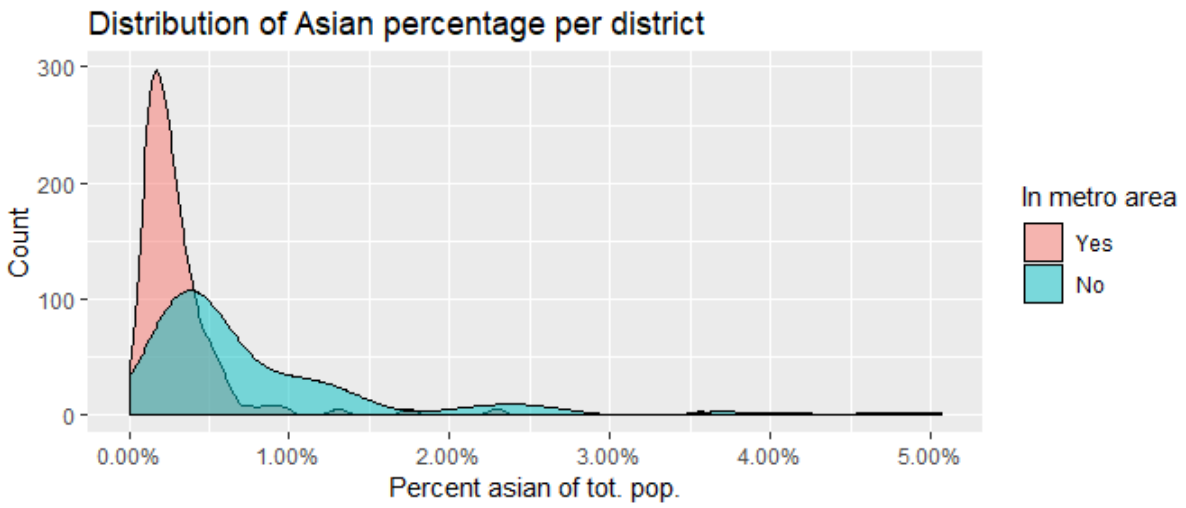
```
midwest %>%
    ggplot(aes(sample=percasian)) + geom_qq() + stat_qq_line()

midwest %>%
    ggplot(aes(sample=log10(percasian))) + geom_qq() + stat_qq_line()
```



Density plot

Verteilung einer numerischen Variable im Vergleich mit wenigen Kategorien

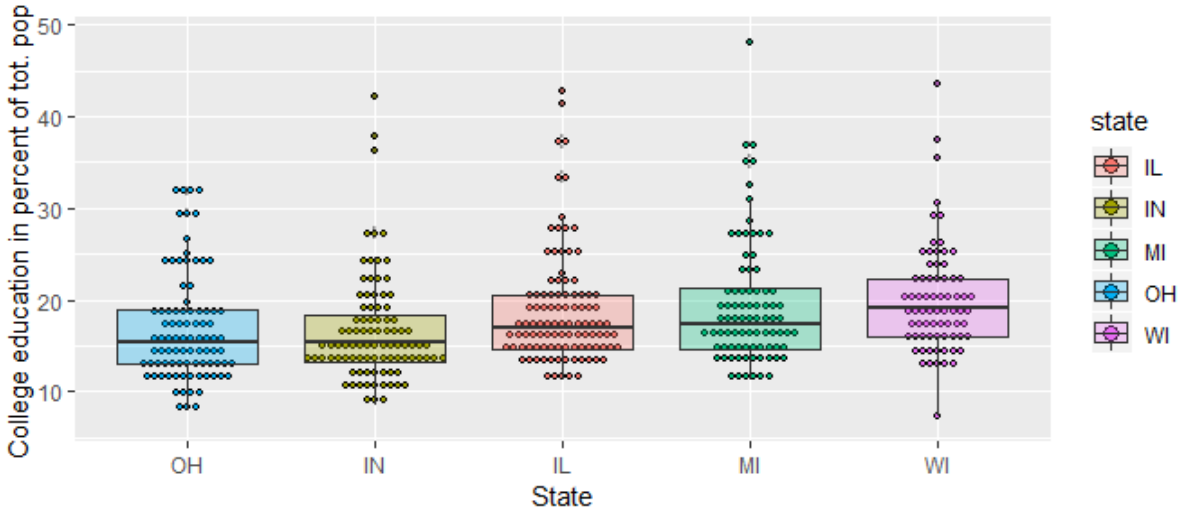
```
midwest %>%
    ggplot(aes(x = percasian / 100, fill = inmetro)) +
    geom_density(alpha=.5) +
    scale_x_continuous(labels = scales::percent) +
    scale_fill_discrete(labels = c('Yes', 'No')) +
    labs(x='Percent asian of tot. pop.', y = 'Count',
        title='Distribution of Asian percentage per district',
        fill = 'In metro area')
```



Boxplot / Dotplot

Verteilung einer numerischen Variable über mehrere Kategorien

```
midwest %>%
    ggplot(aes(x = state %>% fct_reorder(percollege), y = percollege, fill=state)) +
    geom_dotplot(binaxis = 'y', stackdir = 'center', dotsize=.6) +
    geom_boxplot(alpha = .3, outlier.size = 0) +
    labs(x = 'State', y = 'College education in percent of tot. pop.')
```

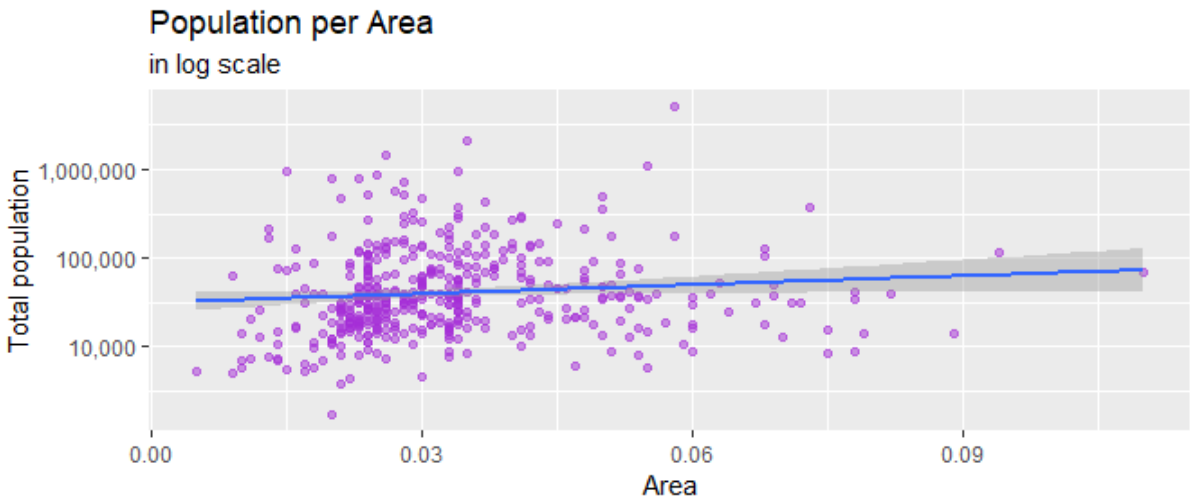


Scatterplot

Relation zwischen zwei numerischen Variablen

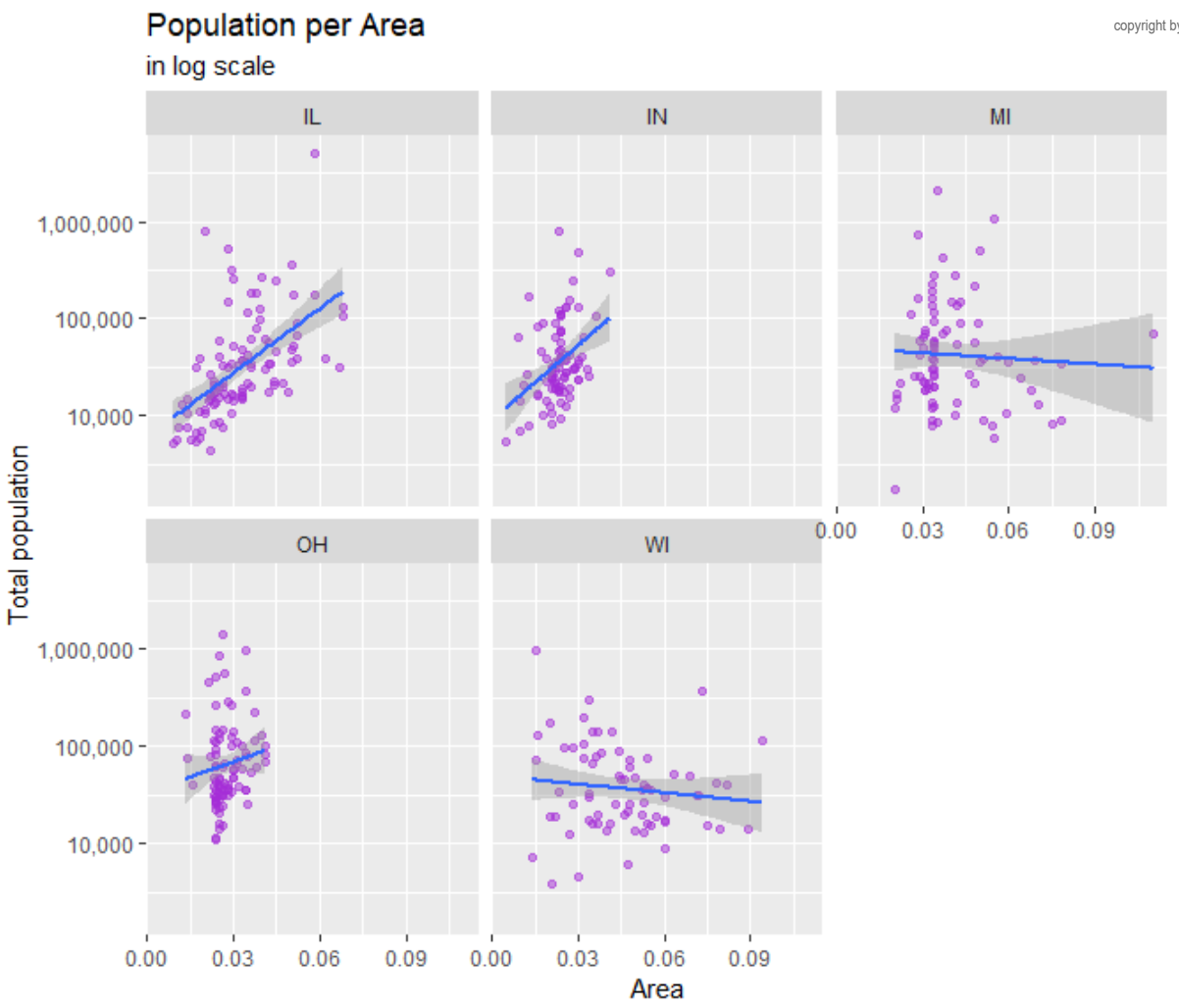
```
midwest %>%
    ggplot(aes(x=area, y=poptotal)) +
```

```
    geom_point(alpha=.5, color='#a52dd7') +
    geom_smooth(method="lm") +
    scale_y_log10(labels= scales::comma) +
    labs(x = 'Area', y = 'Total population',
         title='Population per Area',
         subtitle = 'in log scale')
```



Als *facet-plot* unterschieden nach *state*

```
midwest %>%
    ggplot(aes(x=area, y=poptotal)) +
    geom_point(alpha=.5, color='#a52dd7') +
    geom_smooth(method="lm") +
    scale_y_log10(labels= scales::comma) +
    labs(x = 'Area', y = 'Total population',
         title='Population per Area',
         subtitle = 'in log scale') +
    facet_wrap(vars(state))
```

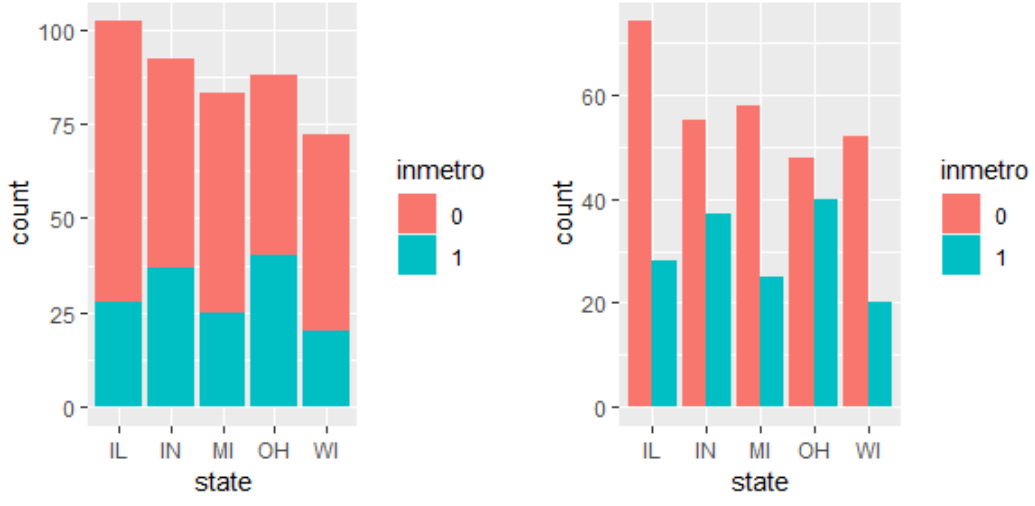


Matrix-Scatterplot

```
library(GGally)

midwest %>%
    select(percollege, percbelowpoverty, percblack, percasian, inmetro) %>%
    GGally::ggpairs(mapping=aes(color=inmetro))
```

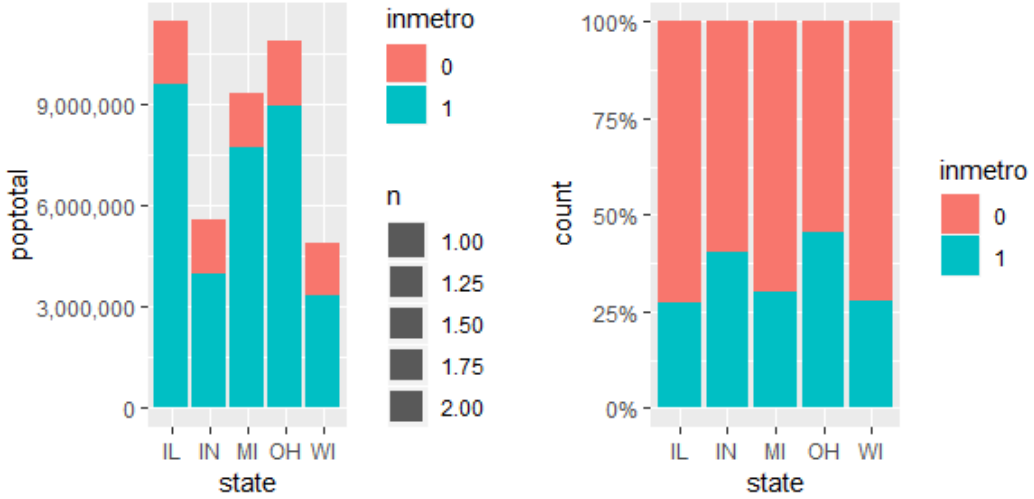
Barplots

```
midwest %>%
    ggplot(aes(state, fill = inmetro)) + geom_bar()

midwest %>%
    ggplot(aes(state, fill = inmetro)) + geom_bar(position='dodge')
```




```
midwest %>%
    ggplot(aes(state, fill = inmetro, y = poptotal)) + geom_bar(stat='sum')
```

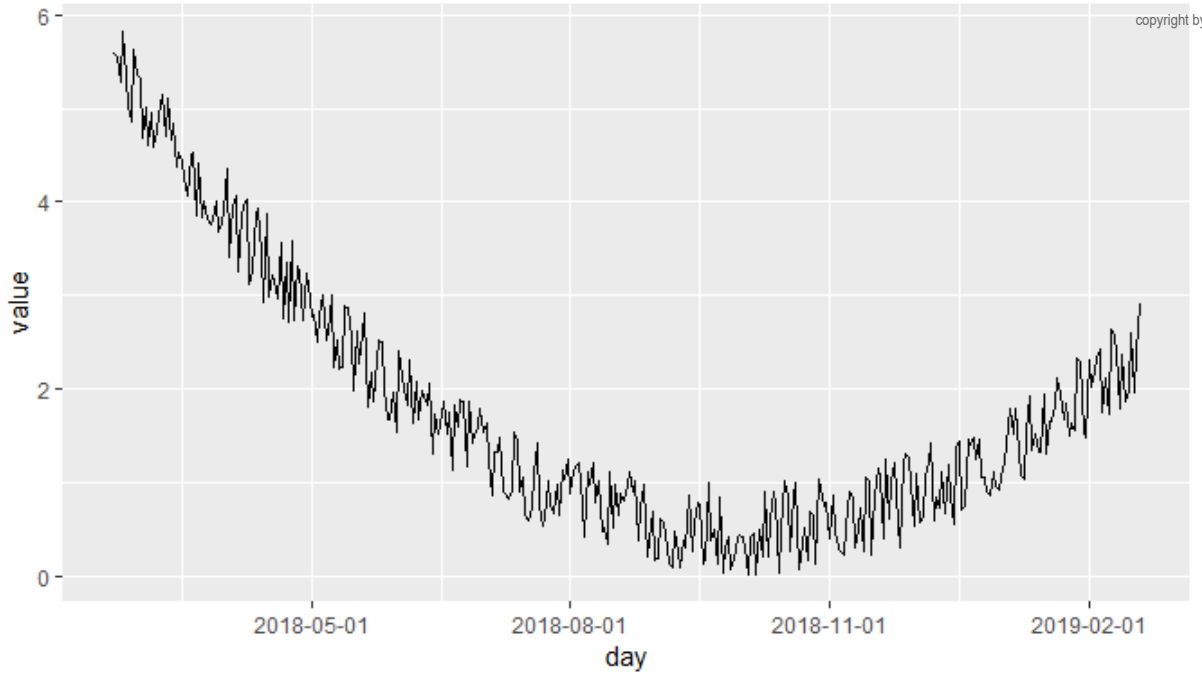
```
    scale_y_continuous(labels=scales::comma)

midwest %>%
    ggplot(aes(state, fill = inmetro)) + geom_bar(position='fill') +
    scale_y_continuous(labels = scales::percent)
```



Timeseries

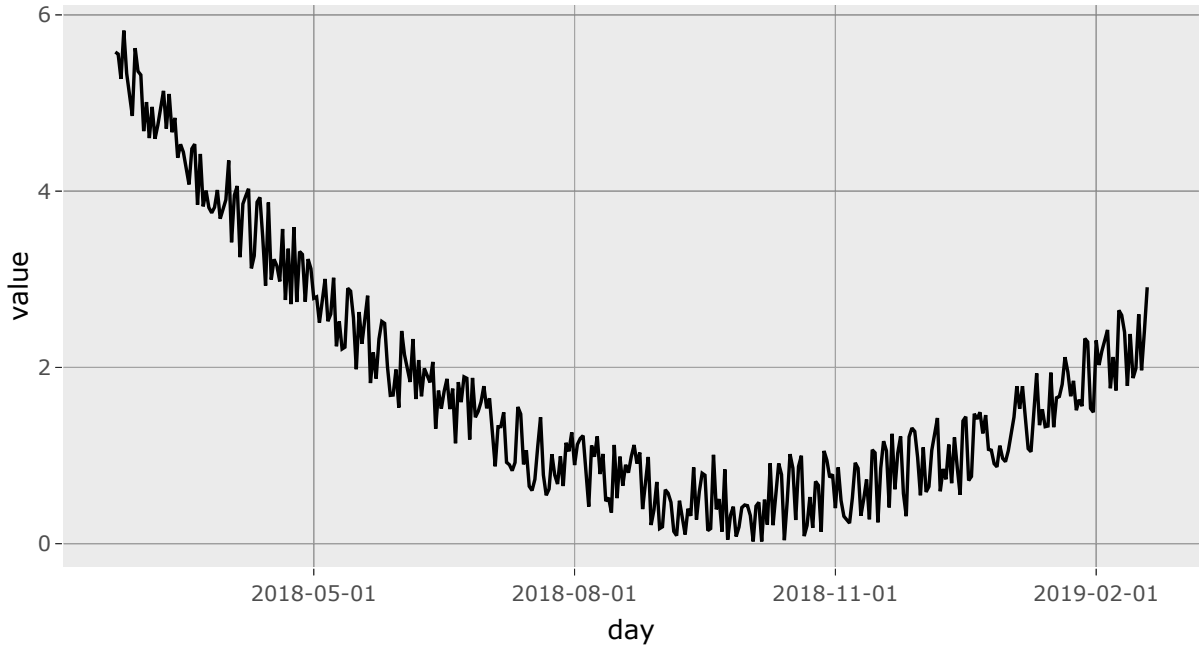
```
# Build a Time series data set
day <- Sys.Date() - 0:364
value <- runif(365) + seq(-140, 224)^2 / 10000
tsdata <- tibble(day, value)
```

```
p <- tsdata %>%
    ggplot(aes(day, value)) +
    geom_line() +
    scale_x_date(
        #date_labels  = '%Y-%m',
        date_breaks = '3 months'
        )
p
```



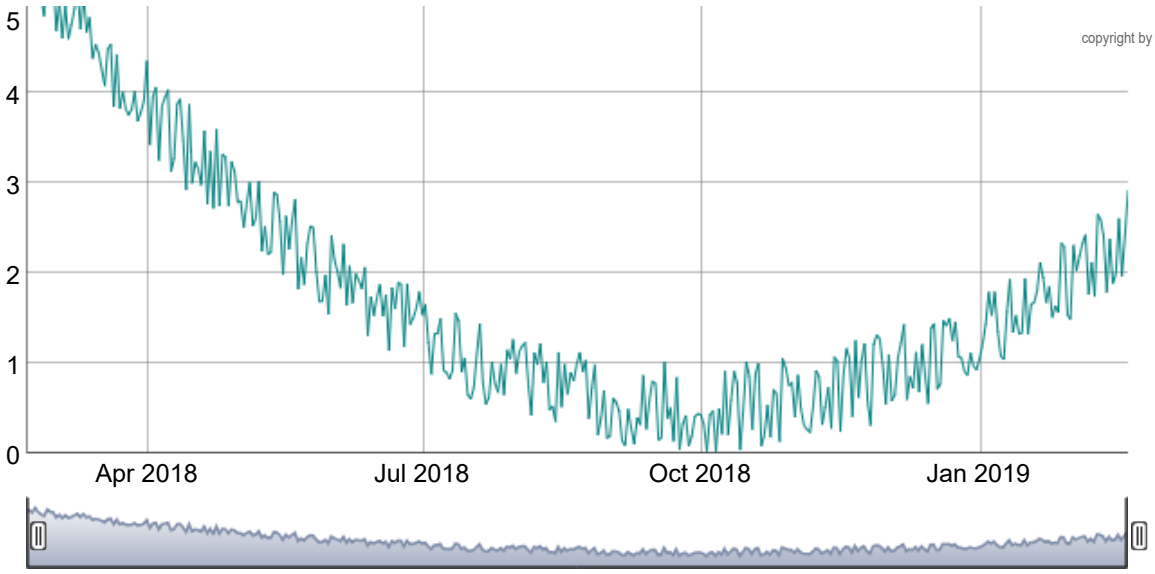
Interaktivität mit `ggplotly`

```
library(plotly)
ggplotly(p)
```



Interaktivität mit `dygraphs`

```
library(dygraphs)

xtsdata <- tsdata %>%
    as.data.frame() %>%
    column_to_rownames("day") %>%
    xts::as.xts()

xtsdata %>% dygraph() %>% dyRangeSelector()
```



Weitere Beispiele

https://www.r-graph-gallery.com/

08 Timeseries Forecast mit prophet

Martin Hanewald

2019-02-19

Packages

```
library(tidyverse)
library(prophet)
library(lubridate)
library(dygraphs)
library(xts)
library(fpp2)
```

Überblick

Zeitreihenvorhersagen mit der `prophet` Bibliothek erzielen gute Ergebnisse mit minimalem Aufwand. Es ist möglich Sondereffekte mit zusätzlichen Regressoren oder Feiertage einzubringen. Prophet wählt automatisch das beste ARIMA Modell mit passender Saisonalität aus. Eine Besonderheit ist die inkludierte Funktion zur Time-Series-Cross-Validation, die ein robusteres Verfahren zur Modellevaluation darstellt als das übliche Splitting in Training- und Testset.

Zentrale Funktionen von `prophet` sind:

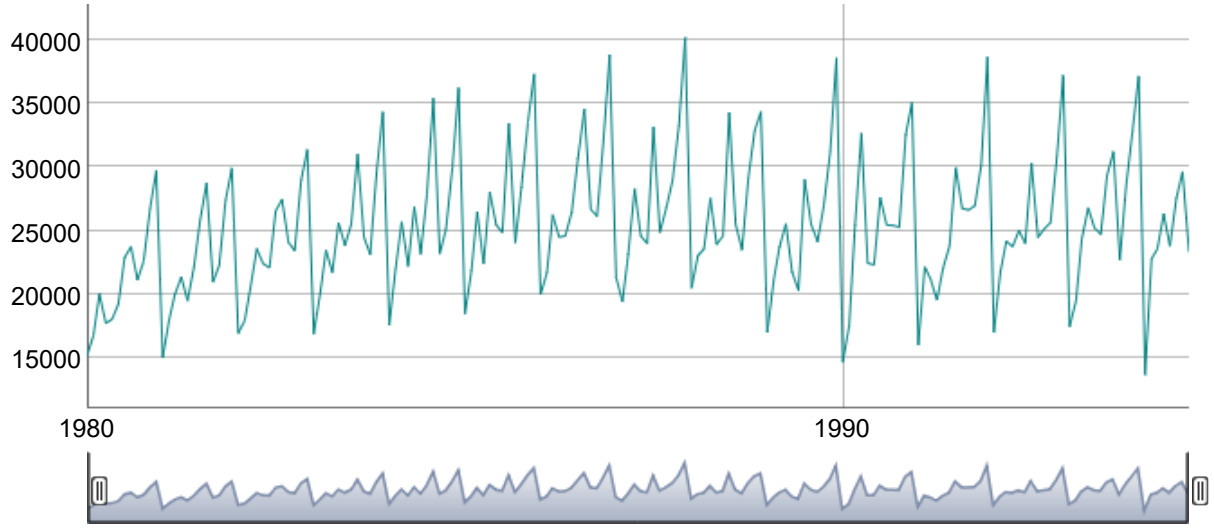
- `prophet()`: Automatisches ARIMA Fitting
- `make_future_dataframe()`: Erzeugung eines DF mit entsprechendem Zukunftshorizont für den Forecast
- `predict()`: Erzeugung des Forecasts
- `dyplot.prophet()`: Schöner interaktiver Plot des Forecasts
- `cross_validation()`: Berechnung der Time-Series Cross Validation
- `plot_cross_validation_metric()`: Plot zur CV

Dataset

Wir verwenden einen Datensatz aus dem `forecast`Paket: `wineind`. Es enthält eine Zeitreihe zum Weinverkauf australischer Winzer zwischen 1980 und 1994. Es eignet sich auf Grund seiner starken saisonalen Struktur sehr gut für einen Forecast.

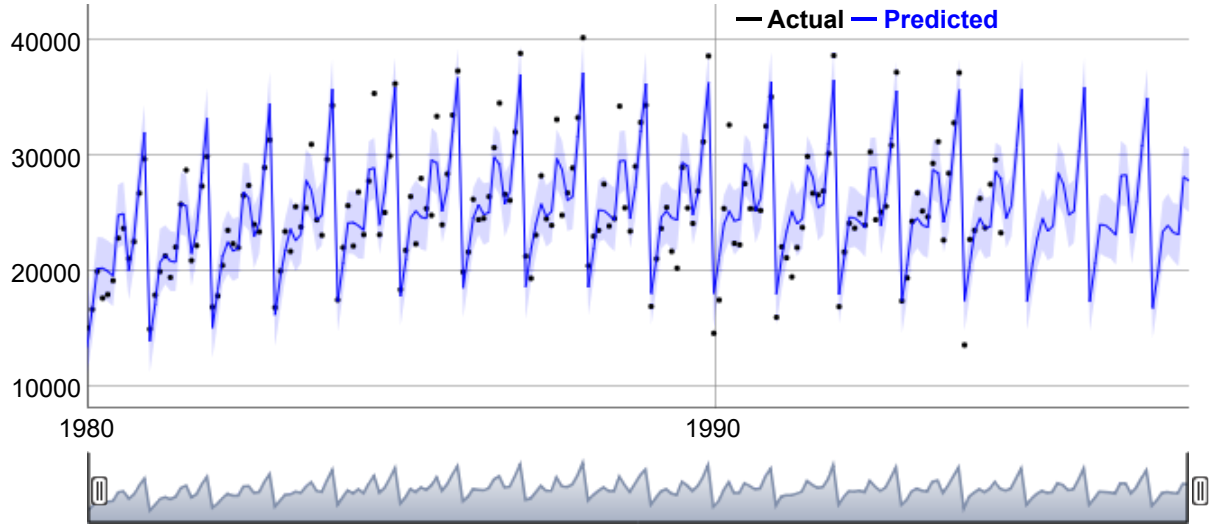
```
data(wineind, package='forecast')
# prophet erwartet einen Dataframe im Format (ds = Datumsspalte, y = Wertespalte)
df <- as_tibble(wineind) %>%
    mutate(ds = time(wineind) %>% as_date()) %>%
    rename(y = x)
```

```
df %>% column_to_rownames('ds') %>%
    dygraph() %>% dygraphs::dyRangeSelector()
#> Warning: Setting row names on a tibble is deprecated.
```

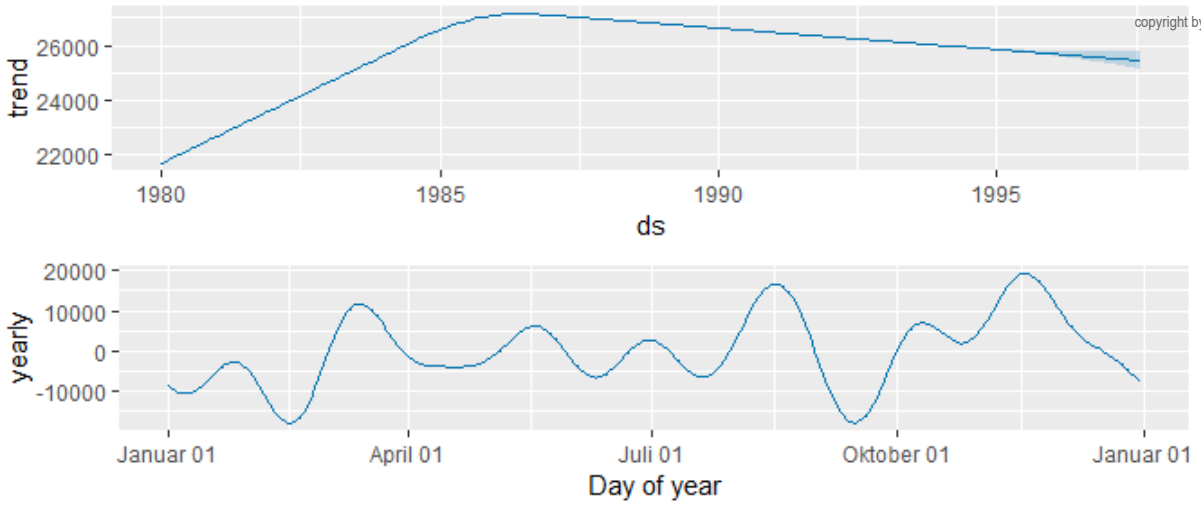


Training

```
m <- prophet::prophet(df)
#> Disabling weekly seasonality. Run prophet with weekly.seasonality=TRUE to override this.
#> Disabling daily seasonality. Run prophet with daily.seasonality=TRUE to override this.
#> Initial log joint probability = -5.6225
#> Optimization terminated normally:
#>   Convergence detected: absolute parameter change was below tolerance

p <- prophet::make_future_dataframe(m, 36, freq = 'month') %>%
    predict(m, .)

prophet::dyplot.prophet(m, p)
```



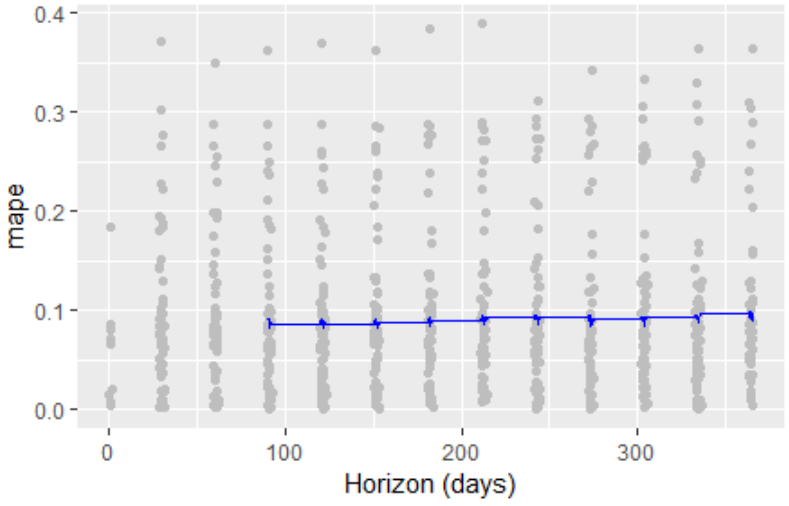

```
# Komponentenplot
prophet::prophet_plot_components(m, p)
```



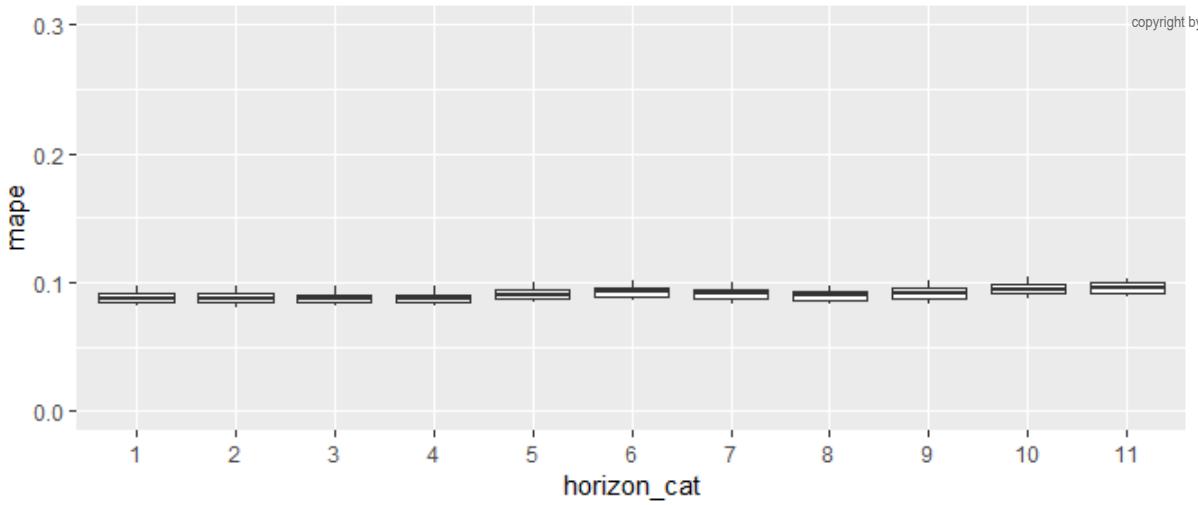
Cross-Validation

```
cv <- prophet::cross_validation(m, horizon = 365, initial=365*8, period = 365/12, units = 'days')
```

```
cv %>% prophet::plot_cross_validation_metric(metric='mape', rolling_window = .2)
```



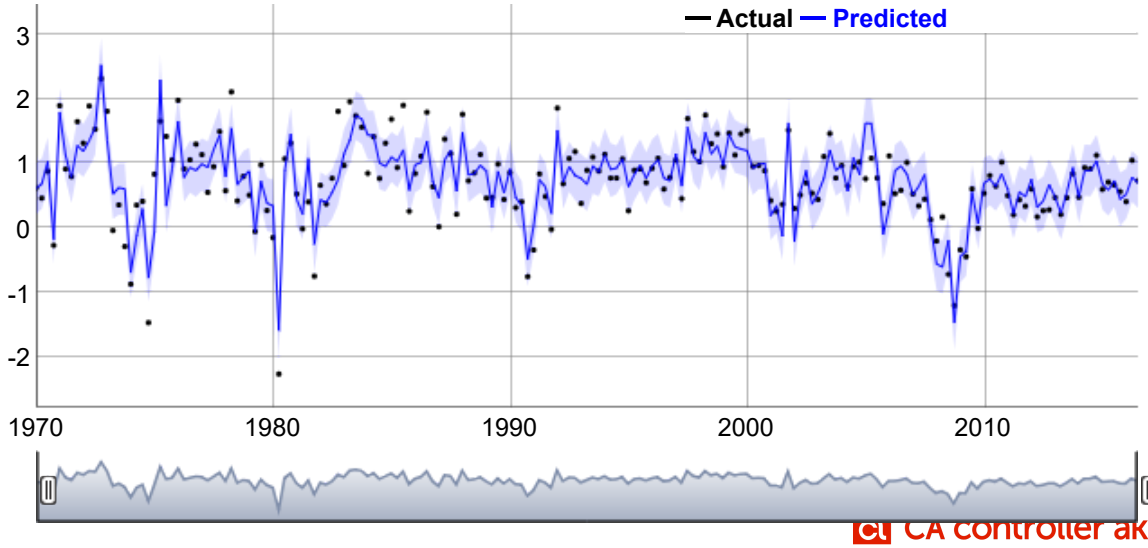

```
cv %>% prophet::performance_metrics() %>% as_tibble() %>%
    mutate(horizon_cat = cut(as.numeric(horizon), 11, labels=1:11)) %>%
    ggplot(aes(x = horizon_cat, y= mape)) + geom_boxplot() + ylim(c(0,.3))
```



Die Cross-Validation zeigt, dass der Forecast Error MAPE im Schnitt leicht unter 10% liegt.

Aufgabe: Führe einen Timeseries Forecast mit zusätzlicher Regression durch

```
data(uschange, package='fpp2')

df <- as_tibble(uschange) %>%
    mutate(ds = time(uschange) %>% as_date) %>%
    rename(y = Consumption)

m <- prophet() %>%
    prophet::add_regressor('Income') %>%
    prophet::add_regressor('Production') %>%
    prophet::add_regressor('Savings') %>%
    prophet::add_regressor('Unemployment') %>%
    fit.prophet(df)
#> Disabling weekly seasonality. Run prophet with weekly.seasonality=TRUE to override this.
#> Disabling daily seasonality. Run prophet with daily.seasonality=TRUE to override this.
#> Initial log joint probability = -9.66493
#> Optimization terminated normally:
#>   Convergence detected: absolute parameter change was below tolerance

p <- df %>%
    predict(m, .)

dyplot.prophet(m, p)
```



09 Machine Learning mit caret

Martin Hanewald

2019-02-19

Packages

```
library(tidyverse)
library(caret)
library(knitr)
library(DT)
library(GGally)
library(Metrics)
```

Überblick

Das `caret` Paket ist der einfachste Weg um eine Vielzahl von Machine-Learning Techniken anzuwenden. Es verfügt über ein einheitliches Interface um zahlreiche Modelle aus anderen Paketen anzusprechen.

Die zentralen FUnktionen sind:

- `createDataPartition()`: Splitting in Training- und Testset
- `train()`: Training von Modellen
- `trainControl()`: Steuerung der Validierungsmethodik und Parametertuning
- `preProcess()`: Optionales Pre-Processing der Daten (Centering, Scaling, etc.)
- `predict()`: Erzeugung der Prediction
- `resamples()`: Erzeugung von Cross-Validation Metriken mit `summary()`und `bwplot()`

Dataset

Wir betrachten einen mitgelieferten Datensatz über den Zusammenhang zwischen Häuserpreisen (`price`) und verschiedenen Hauseigenschaften wie Größe, Lage, Anzahl Badezimmer, etc.

```
data(Sacramento, package='caret')
Sacramento <- Sacramento %>% as_tibble()
# für scrollbaren HTML Output
Sacramento %>% sample_n(20) %>% datatable(options = list(scrollX=T))
```

Show 10 entries Search:

	city	zip	beds	baths	sqft	type	price	latitude
1	GOLD_RIVER	z95670	2	2	1520	Residential	299000	38.62869
2	CITRUS_HEIGHTS	z95621	4	2	1280	Residential	167293	38.715781
3	SACRAMENTO	z95864	3	1	1643	Residential	99000	38.588672
4	GALT	z95632	5	4	3746	Residential	420000	38.271646
5	SACRAMENTO	z95826	3	2	1280	Residential	192067	38.560767
6	SACRAMENTO	z95842	4	2	1578	Residential	195000	38.693071
7	SACRAMENTO	z95841	3	2	1120	Residential	178000	38.658734

8	SACRAMENTO	z95815	2	1	1011	Residential	85000	38.6238
9	SACRAMENTO	z95818	2	1	1144	Residential	299000	38.556844
10	CITRUS_HEIGHTS	z95621	3	2	1343	Residential	284893	38.715853

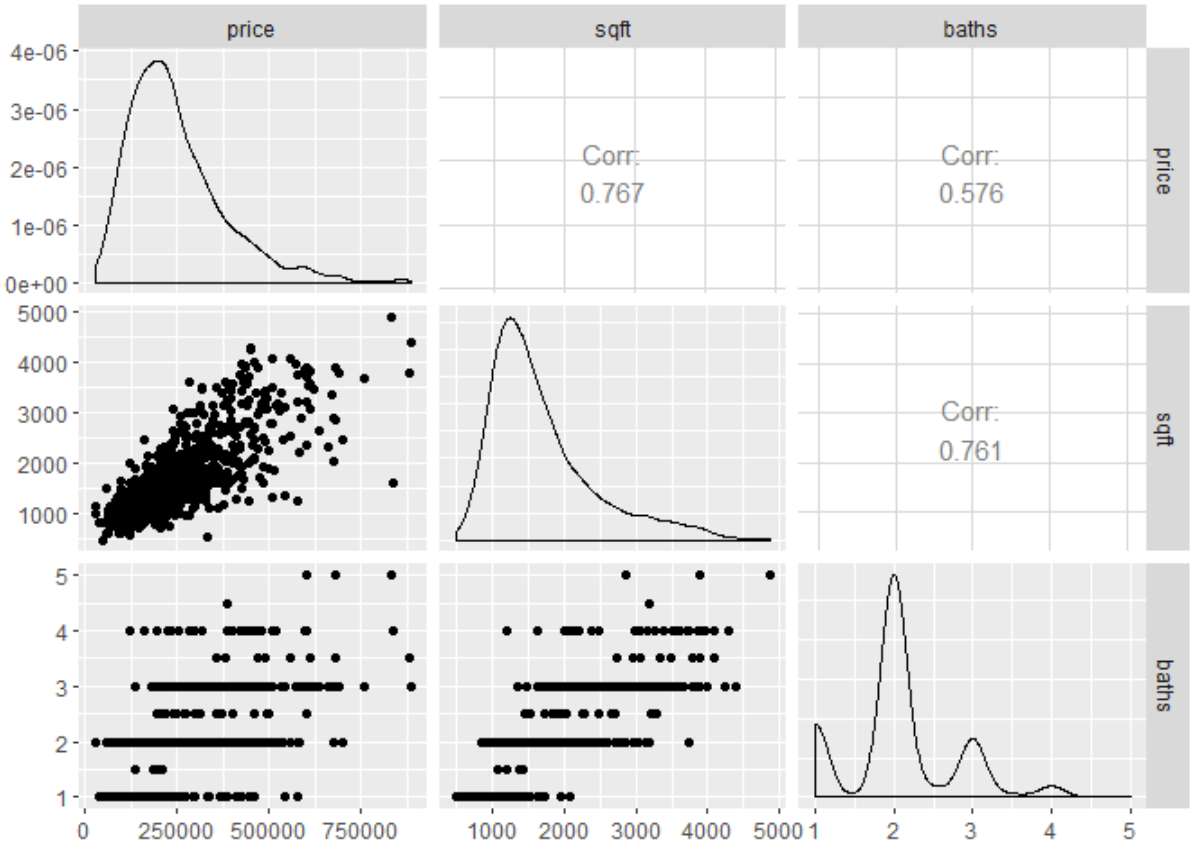
Showing 1 to 10 of 20 entries

Previous 1 2 Next

Exploration

Ein einfacher Pairs Plot zwischen zwei ausgewählten Variabeln `sqft` und `baths` zeigt eine deutliche positivte Korrelation.

```
Sacramento %>% select(price, sqft, baths) %>% ggpairs()
```



Splitting und Training

Wir splitten den Datensatz gemäß der 80/20 Faustregel.

```
in_train <- createDataPartition(Sacramento$price, p = .8, list = FALSE)

training <- Sacramento[ in_train,]
testing  <- Sacramento[-in_train,]
```

Danach fitten wir ein einfaches lineares Modell sowie einen Decision Tree.

Als Sampling Methode wählen wir die Cross-Validation (method='cv') und einige Pre-Processing Schritte für numerische Variablen. Als einzigen Prädiktor wählen wir zunächst nur die Variable `sqft`.

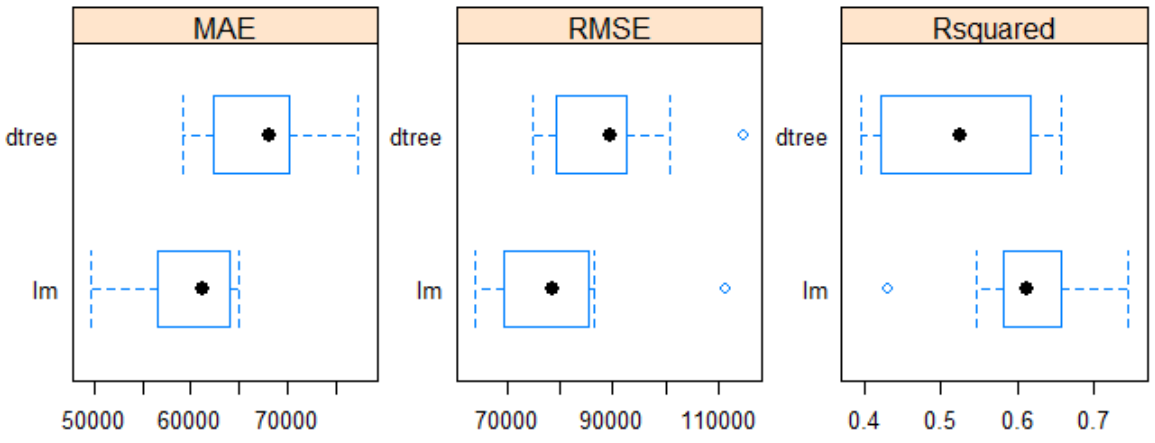
```
fit_lm <- train(price ~ sqft,
             trControl = trainControl(method='cv'),
             preProcess = c('center', 'scale', 'zv'),
             method = 'lm',
             data=training)
```

```
fit_dtree <- train(price ~ sqft,
             trControl = trainControl(method='cv'),
             preProcess = c('center', 'scale', 'zv'),
             method = 'rpart',
             data=training)
#> Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info =
#> trainInfo, : There were missing values in resampled performance measures.
```

Evaluation

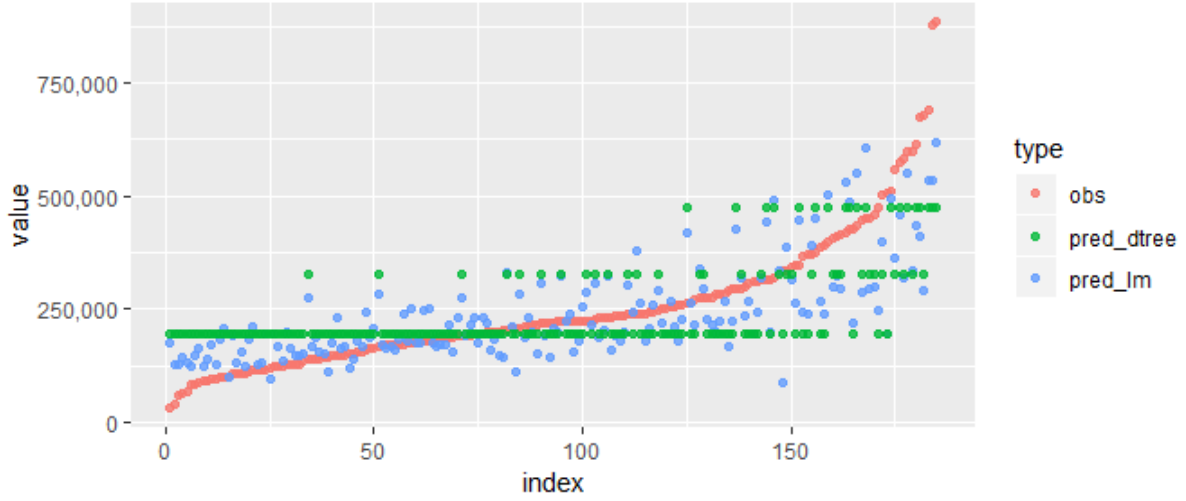
Die Evaluation anhand der Kreuzvalidierungssamples zeigt das lineare Modell eindeutig im Vorteil.

```
res <- resamples(list(lm = fit_lm, dtree = fit_dtree))
res %>% summary()
#>
#> Call:
#> summary.resamples(object = .)
#>
#> Models: lm, dtree
#> Number of resamples: 10
#>
#> MAE
#>           Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
#> lm    49662.66 56608.57 61138.46 59482.88 63998.12 64976.63    0
#> dtree 59116.95 62521.25 68131.29 67158.75 69759.41 77328.92    0
#>
#> RMSE
#>           Min. 1st Qu.   Median     Mean  3rd Qu.     Max. NA's
#> lm    63948.38 70316.6 78445.73 79636.29 85436.34 111279.2    0
#> dtree 74841.75 79334.8 89396.70 88917.08 92501.02 114739.1    0
#>
#> Rsquared
#>            Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
#> lm    0.4305718 0.5831811 0.6119248 0.6102292 0.6510016 0.7443224    0
#> dtree 0.3947227 0.4225706 0.5244397 0.5151145 0.6057425 0.6563568    0
res %>% bwplot(scales='free')
```



Auch die visuelle Inspektion der Prediction auf Basis des Testdatensatzes zeigt die Schwächen des Decision Tree Ansatzes.

```
ans <- tibble(obs = testing$price,
              pred_lm = predict(fit_lm, testing),
              pred_dtree = predict(fit_dtree, testing))

ans %>% arrange(obs) %>%
    mutate(index = row_number(obs)) %>%
    gather(type, value, -index) %>%
    ggplot(aes(x=index, y =value, color=type)) + geom_point(alpha=.8) +
    scale_y_continuous(labels = scales::comma)
```



Abschließende Beurteilung der Güte auf dem Testset anhand der Performance-Kennzahl MAPE (Mean Absolute Percentage Error).

```
ans %>%
    summarise(mape_lm = mape(obs, pred_lm),
              mape_dtree = mape(obs, pred_dtree),
              mase_lm = mase(obs, pred_lm),
              mase_dtree = mase(obs, pred_dtree))
#> # A tibble: 1 x 4
#>   mape_lm mape_dtree mase_lm mase_dtree
#>     <dbl>      <dbl>   <dbl>      <dbl>
#> 1   0.320      0.415    2.52       2.92
```

Aufgabe: Verbessere obiges Modell durch Variation

1. Probiere ein Ensemble Modell aus (siehe caret Doku)
2. Füge zusätzliche Variablen hinzu

10 Fallstudie: Churn Prediction

Martin Hanewald

2019-02-19

Packages

Überblick

Dataset

Zwei Datensätze:

- User-Stammdaten
- Aktivitäten pro User

```
activity_raw <-  read_csv('data/activity_info.csv')
user_raw <- read_csv('data/user_info.csv')
```

Pre-Processing

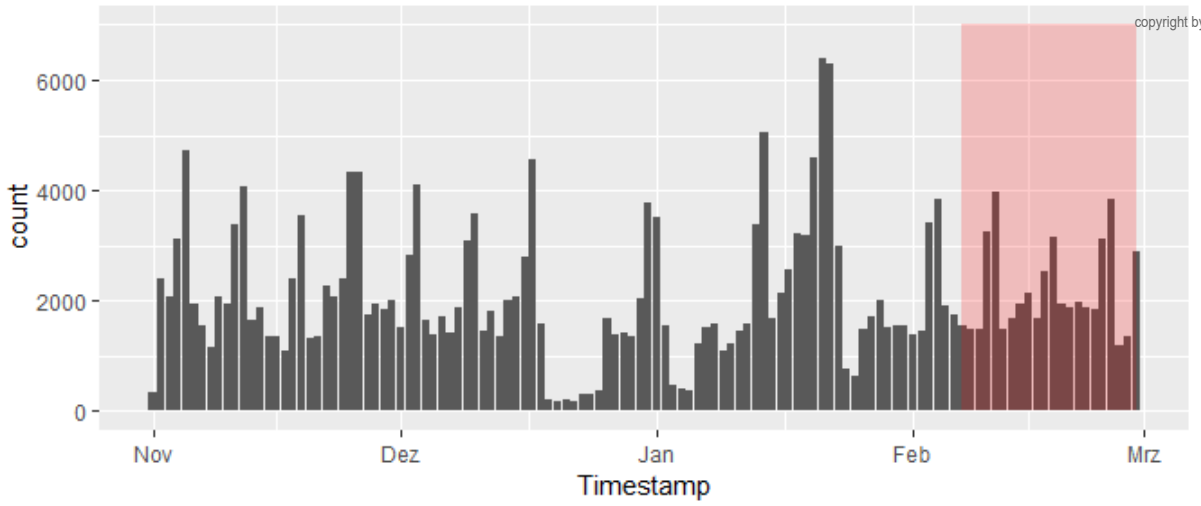
```
CHURN_PERIOD <- 21
CHURN_THRESHOLD <- 0
```

Cleanup

```
activity <- activity_raw %>%
    mutate(Timestamp = mdy_hm(Timestamp) %>% as_date() + years(17)) %>%
    mutate(ItemId = str_pad(ItemId, 12, pad='0')) %>%
    mutate_at(vars(TransactionId, UserId, ItemId), as.factor) %>%
    select(-Location, -ProductCategory, -X1)

user <- user_raw %>%
    select(-Gender, -UserType) %>%
    mutate_all(as.factor)
```

Create the `churn` label

```
last_date <- max(activity$Timestamp)

# Show Plot with churn period
activity %>%
    ggplot(aes(Timestamp)) + geom_bar() +
    annotate(geom='rect',
             xmin=last_date-CHURN_PERIOD,
             ymin = 0, ymax=7000, xmax=last_date,
             fill='red', alpha=.2)
```




```
# Approach one: For-loop, save results in list
churn <- list()

for(u in unique(activity$UserId)){
    churn[[u]] <- activity %>%
        filter(UserId == u) %>%
        filter(Timestamp > last_date - CHURN_PERIOD) %>%
        nrow() <= CHURN_THRESHOLD
}

churn_label <- churn %>% as_tibble() %>%
    gather(UserId, churn)

# profiler: 25,56 sec
```

```
# Approach two: Group_by and summarise
churn_label <- activity %>%
    mutate(in_period = Timestamp > last_date - CHURN_PERIOD) %>%
    select(UserId, TransactionId, in_period) %>% unique() %>%
    group_by(UserId) %>%
    summarise(churn = sum(in_period) <= CHURN_THRESHOLD) %>%
    mutate_at(vars(churn), as.factor)

# profiler: 3,6 sec

churn_label$churn %>% table()
#> .
#> FALSE  TRUE
#>  4492  5508
```

Feature Engineering

- Total_Quantity
- Total_Value
- StDev_Quantity
- StDev_Value
- AvgTimeDelta
- Recency
- Count_Unique_TransactionId
- Count_Unique_ItemId

- Mean_Quantity_per_Unique_TransactionId
- Mean_Quantity_per_Unique_ItemId
- Mean_Value_per_Unique_TransactionId
- Mean_Value_per_Unique_ItemId

```
activity_before <- activity %>% filter(Timestamp < last_date - CHURN_PERIOD)

activity_measures <-
    activity_before %>%
        group_by(UserId) %>%
        arrange(Timestamp, .by_group=TRUE) %>%
        summarise(Total_Quantity = sum(Quantity),
                  Total_Value = sum(Value),
                  StDev_Quantity = sd(Quantity),
                  StDev_Value = sd(Value),
                  AvgTimeDelta = mean(diff(Timestamp)),
                  Recency = last_date - CHURN_PERIOD - max(Timestamp)) %>%
    full_join(
        activity_before %>%
            group_by(UserId, ItemId) %>%
            summarise(Quantity_sum = sum(Quantity),
                      Value_sum = sum(Value)) %>%
            summarise(Mean_Quantity_per_Unique_ItemId = mean(Quantity_sum),
                      Mean_Value_per_Unique_ItemId = mean(Value_sum),
                      Count_Unique_ItemId = n_distinct(ItemId))) %>%
    full_join(
        activity_before %>%
            group_by(UserId, TransactionId) %>%
            summarise(Quantity_sum = sum(Quantity),
                      Value_sum = sum(Value)) %>%
            summarise(Mean_Quantity_per_Unique_TransactionId = mean(Quantity_sum),
                      Mean_Value_per_Unique_TransactionId = mean(Value_sum),
                      Count_Unique_TransactionId = n_distinct(TransactionId)))
#> Joining, by = "UserId"
#> Joining, by = "UserId"
```

Bind final model dataframe

```
modeldat <- user %>%
    inner_join(churn_label) %>%
    inner_join(activity_measures) %>%
    drop_na() # 724 lines dropped where only one transaction per user
#> Joining, by = "UserId"
#> Joining, by = "UserId"

# check for na's
modeldat %>%
    summarise_all(function(x) is.na(x) %>% sum()) %>%
    gather(var, na) %>% filter(na>0)
#> # A tibble: 0 x 2
#> # ... with 2 variables: var <chr>, na <int>
```

Modelling

Splitting

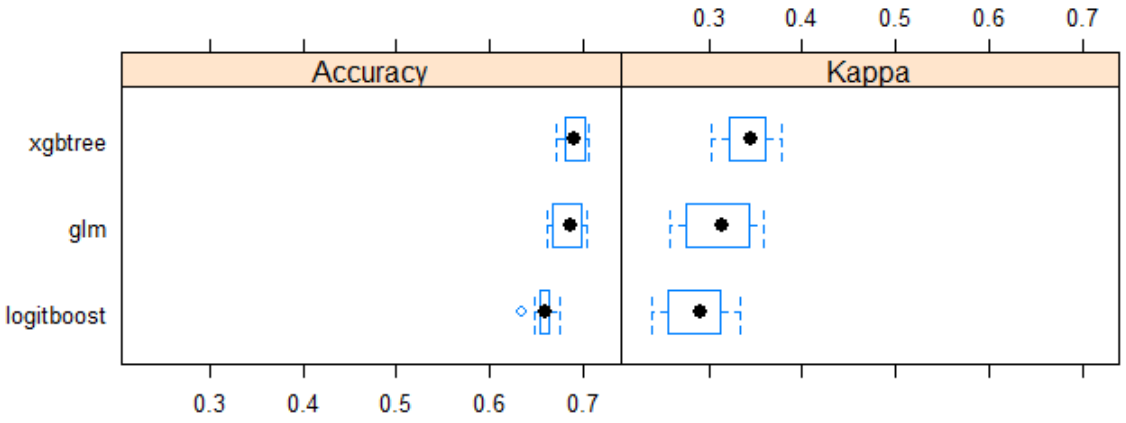
```
inTrain <- createDataPartition(modeldat$churn, p = .8, list=F)
training <- modeldat[inTrain,]
testing <- modeldat[-inTrain,]
```

Training

```
trControl <- trainControl(method='cv')

fit <- list()

fit$glm <- train(churn ~ . -UserId,
                 method = 'glm',
                 family = 'binomial',
                 trControl = trControl,
                 preProcess = c('center', 'scale'),
                 data = training)

fit$logitboost <- train(churn ~ . -UserId,
                 method = 'LogitBoost',
                 trControl = trControl,
                 preProcess = c('center', 'scale'),
                 data = training)

fit$xgbtree <- train(churn ~ . -UserId,
                 method = 'xgbTree',
                 trControl = trControl,
                 preProcess = c('center', 'scale'),
                 data = training)
```

Evaluation

```
resamples(fit) %>% bwplot()
```




```
varImp(fit$glm)
#> glm variable importance
#>
#>   only 20 most important variables shown (out of 29)
#>
```

```
#>                                   Overall
#> Recency                          100.000
#> AvgTimeDelta                      68.117
#> AddressE                          43.496
#> AddressF                          33.461
#> Mean_Value_per_Unique_TransactionId  28.174
#> Count_Unique_TransactionId        27.860
#> AddressD                          27.566
#> StDev_Value                       22.180
#> AddressH                          21.786
#> AddressG                          18.899
#> AgeJ                              14.706
#> Mean_Value_per_Unique_ItemId      13.378
#> AddressC                          11.109
#> Total_Quantity                    10.964
#> Total_Value                       10.604
#> AgeC                               9.446
#> AddressB                           7.652
#> AgeB                               5.492
#> AgeI                               4.946
#> AgeK                               4.720
#varImp(fit$logitboost)
varImp(fit$xgbtree)
#> xgbTree variable importance
#>
#>   only 20 most important variables shown (out of 29)
#>
#>                                     Overall
#> AvgTimeDelta                      100.0000
#> Count_Unique_TransactionId         33.7933
#> Recency                            19.8737
#> Total_Quantity                     16.2686
#> StDev_Value                         7.3689
#> Count_Unique_ItemId                 7.0082
#> Mean_Value_per_Unique_TransactionId  5.4638
#> Mean_Value_per_Unique_ItemId        5.4284
#> Total_Value                         4.2315
#> Mean_Quantity_per_Unique_TransactionId  3.4107
#> AddressE                            3.3424
#> Mean_Quantity_per_Unique_ItemId     3.2781
#> StDev_Quantity                      1.8020
#> AddressC                            1.3987
#> AgeJ                                0.8408
#> AgeC                                0.3751
#> AgeE                                0.2329
#> AddressB                            0.2147
#> AddressF                            0.1489
#> AgeD                                0.0000

fit$glm %>% predict(testing) %>% confusionMatrix(testing$churn)
#> Confusion Matrix and Statistics
#>
#>           Reference
#> Prediction FALSE TRUE
#>      FALSE   336  172
#>      TRUE    372  849
#>
#>                Accuracy : 0.6854
#>                  95% CI : (0.6629, 0.7072)
#>     No Information Rate : 0.5905
#>     P-Value [Acc > NIR] : 2.359e-16
#>
#>                   Kappa : 0.32
#>  Mcnemar's Test P-Value : < 2.2e-16
```

```
#>
#>             Sensitivity : 0.4746
#>             Specificity : 0.8315
#>          Pos Pred Value : 0.6614
#>          Neg Pred Value : 0.6953
#>              Prevalence : 0.4095
#>          Detection Rate : 0.1943
#>    Detection Prevalence : 0.2938
#>       Balanced Accuracy : 0.6531
#>
#>        'Positive' Class : FALSE
#>
fit$logitboost %>% predict(testing) %>% confusionMatrix(testing$churn)
#> Confusion Matrix and Statistics
#>
#>           Reference
#> Prediction FALSE TRUE
#>      FALSE   432  305
#>      TRUE    276  716
#>
#>                Accuracy : 0.664
#>                  95% CI : (0.6411, 0.6862)
#>     No Information Rate : 0.5905
#>     P-Value [Acc > NIR] : 1.989e-10
#>
#>                   Kappa : 0.3095
#>  Mcnemar's Test P-Value : 0.2454
#>
#>             Sensitivity : 0.6102
#>             Specificity : 0.7013
#>          Pos Pred Value : 0.5862
#>          Neg Pred Value : 0.7218
#>              Prevalence : 0.4095
#>          Detection Rate : 0.2499
#>    Detection Prevalence : 0.4263
#>       Balanced Accuracy : 0.6557
#>
#>        'Positive' Class : FALSE
#>
fit$xgbtree %>% predict(testing) %>% confusionMatrix(testing$churn)
#> Confusion Matrix and Statistics
#>
#>           Reference
#> Prediction FALSE TRUE
#>      FALSE   378  206
#>      TRUE    330  815
#>
#>                Accuracy : 0.69
#>                  95% CI : (0.6676, 0.7117)
#>     No Information Rate : 0.5905
#>     P-Value [Acc > NIR] : < 2.2e-16
#>
#>                   Kappa : 0.3413
#>  Mcnemar's Test P-Value : 1.08e-07
#>
#>             Sensitivity : 0.5339
#>             Specificity : 0.7982
#>          Pos Pred Value : 0.6473
#>          Neg Pred Value : 0.7118
#>              Prevalence : 0.4095
#>          Detection Rate : 0.2186
#>    Detection Prevalence : 0.3378
#>       Balanced Accuracy : 0.6661
#>
```

```
#>         'Positive' Class : FALSE
#>
```