

Gruppe 4 - Final

Daten einlesen

Bereitgestellte Daten einlesen (im Hintergrund)

Feiertagsdaten aus dem Netz holen (im Hintergrund)

Daten zusammenfügen

```
## # A tibble: 10,899 x 11
##   Datum      Warengruppe Umsatz Wochentag KielerWoche Bewoelkung Temperatur
##   <date>      <fct>      <dbl> <fct>      <dbl>      <dbl>      <dbl>
## 1 2013-07-01 Brot          149. Montag          NA           6        17.8
## 2 2013-07-02 Brot          160. Dienstag         NA           3        17.3
## 3 2013-07-03 Brot          112. Mittwoch          NA           7        21.1
## 4 2013-07-04 Brot          169. Donnerstag         NA           7        18.8
## 5 2013-07-05 Brot          171. Freitag           NA           5        20.0
## 6 2013-07-06 Brot          175. Samstag           NA           0        19.0
## 7 2013-07-07 Brot           92.6 Sonntag           NA           0        21.4
## 8 2013-07-08 Brot          136. Montag           NA           0        22.7
## 9 2013-07-09 Brot          136. Dienstag         NA           0        23.3
## 10 2013-07-10 Brot          135. Mittwoch          NA           2        19.7
## # ... with 10,889 more rows, and 4 more variables: Windgeschwindigkeit <dbl>,
## #   Wettercode <dbl>, Feiertage <chr>, istFeiertag <dbl>
```

Datenaufbereitung

Datenstruktur untersuchen

Data summary

Name	dataset
Number of rows	10899
Number of columns	11
Column type frequency:	
character	1
Date	1
factor	2
numeric	7

Group variables

None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
Feiertage	10784	0.01	11	25	0	5	0








Variable type: Date

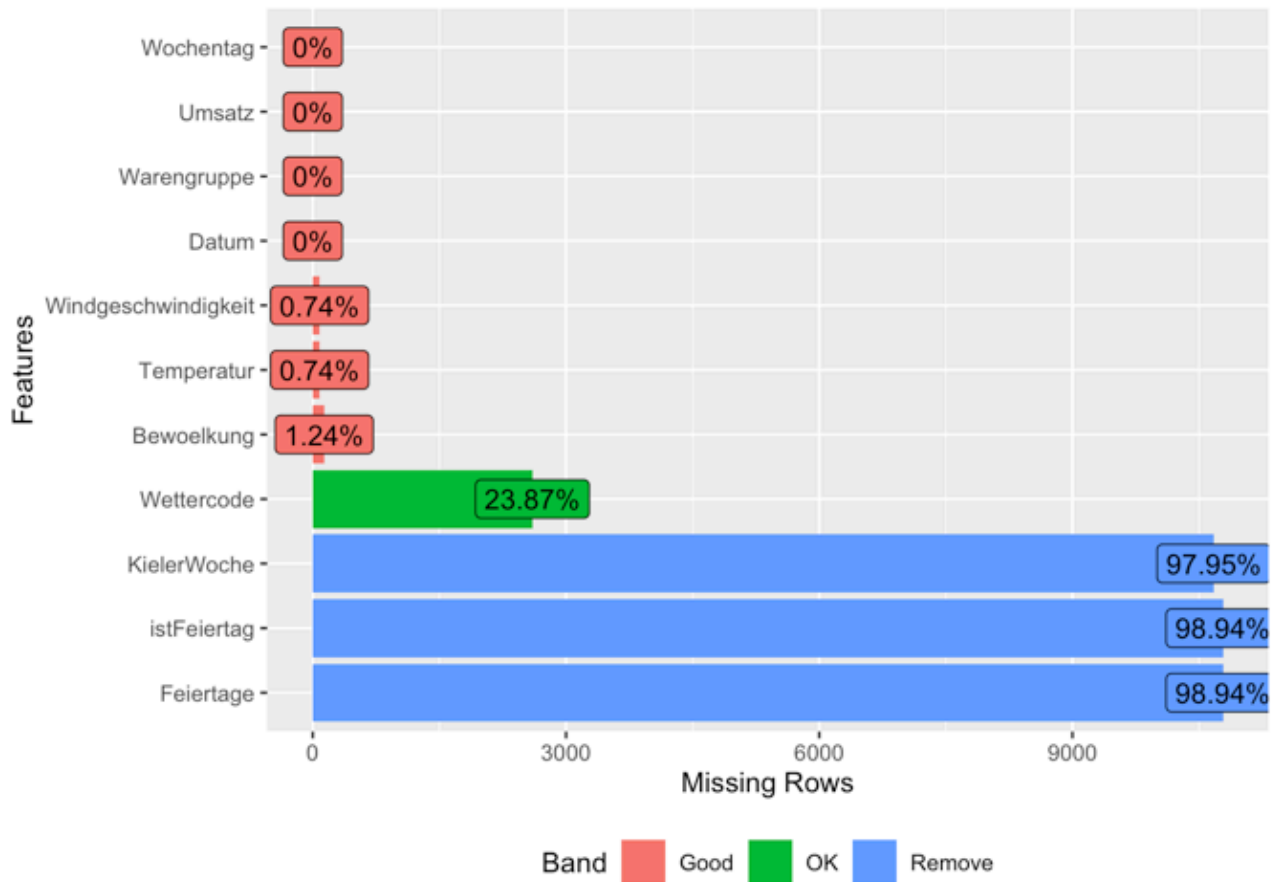
skim_variable	n_missing	complete_rate	min	max	median	n_unique
Datum	0	1	2013-07-01	2019-06-06	2016-06-17	2121

Variable type: factor

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
Warengruppe	0	1	FALSE	6	Bro: 2121, Bro: 2121, Cro: 2121, Kuc: 2121
Wochentag	0	1	FALSE	7	Son: 1570, Don: 1568, Sam: 1564, Die: 1562

Variable type: numeric

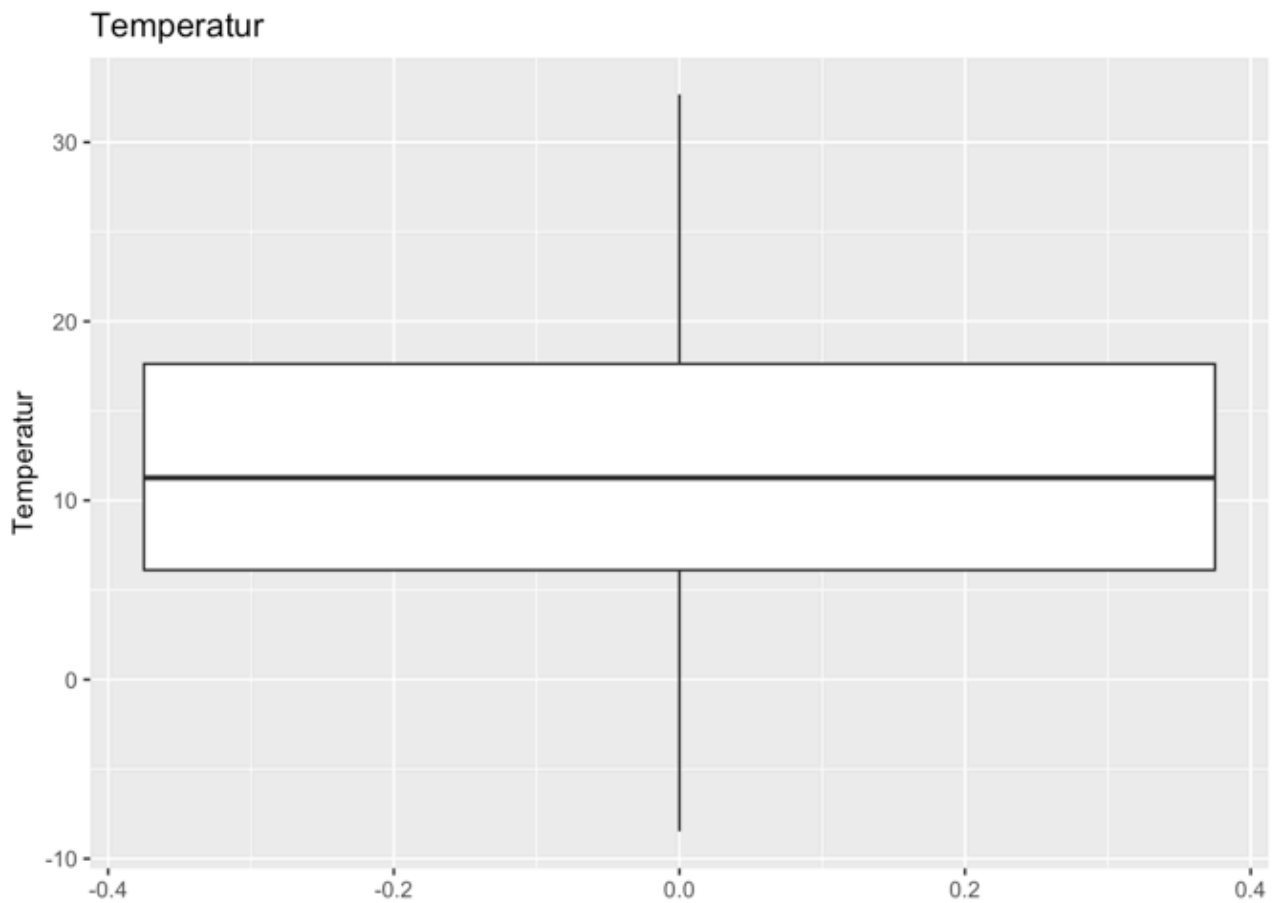
skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
Umsatz	0	1.00	206.66	142.81	7.05	97.53	163.30	280.81	1879.46	
KielerWoche	10676	0.02	1.00	0.00	1.00	1.00	1.00	1.00	1.00	
Bewoelkung	135	0.99	4.73	2.65	0.00	3.00	6.00	7.00	8.00	
Temperatur	81	0.99	11.82	7.15	-8.47	6.11	11.26	17.62	32.67	
Windgeschwindigkeit	81	0.99	11.00	4.14	3.00	8.00	10.00	13.00	35.00	
Wettercode	2602	0.76	36.16	27.04	0.00	10.00	22.00	61.00	95.00	
istFeiertag	10784	0.01	1.00	0.00	1.00	1.00	1.00	1.00	1.00	



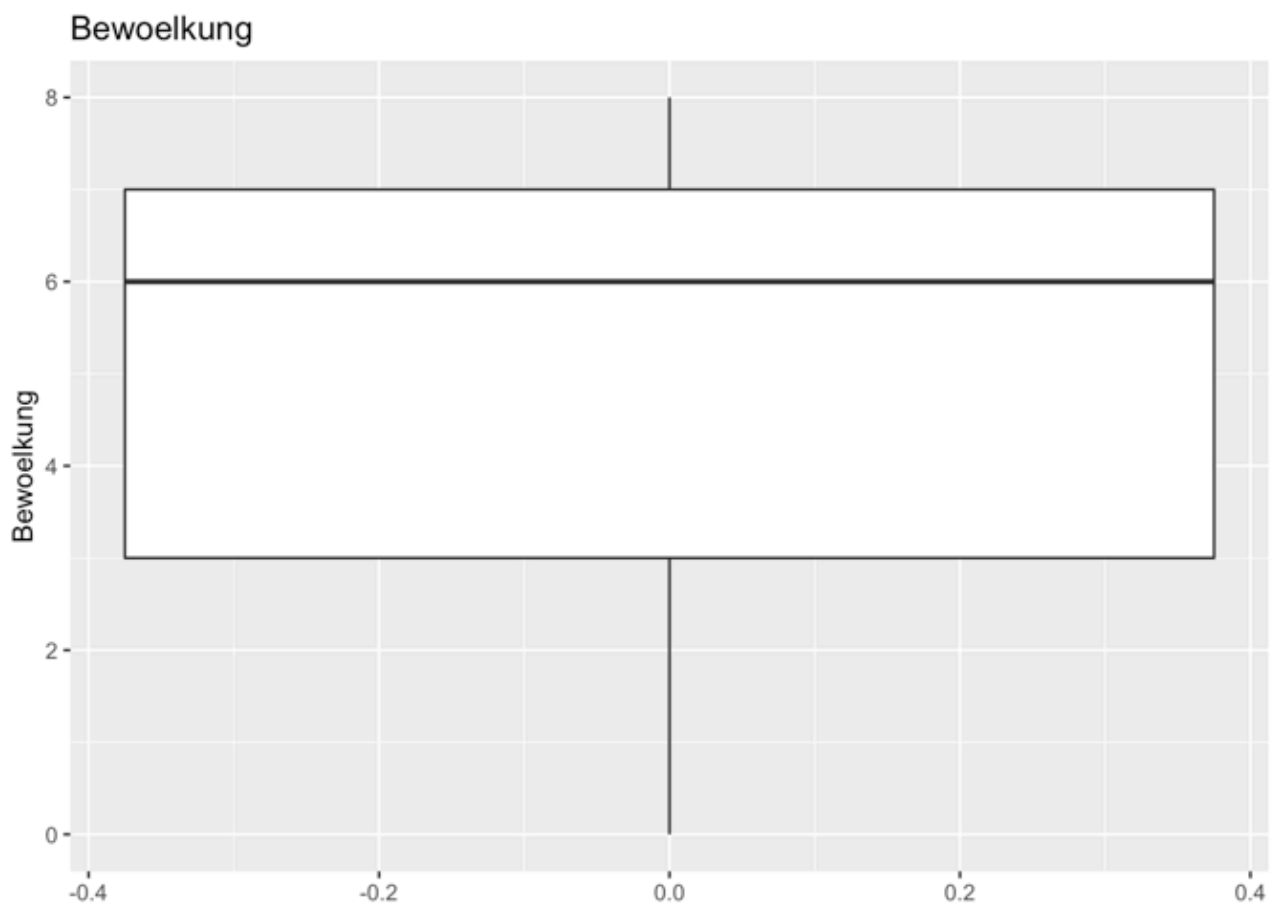
Analyse der Ausreißer in den metrischen Variablen

- Temperatur: keine
- Bewoelkung: keine
- Windgeschwindigkeit: Ausreißer
- Umsatz: Ausreißer
- Umsatz nach Warengruppen: Ausreißer

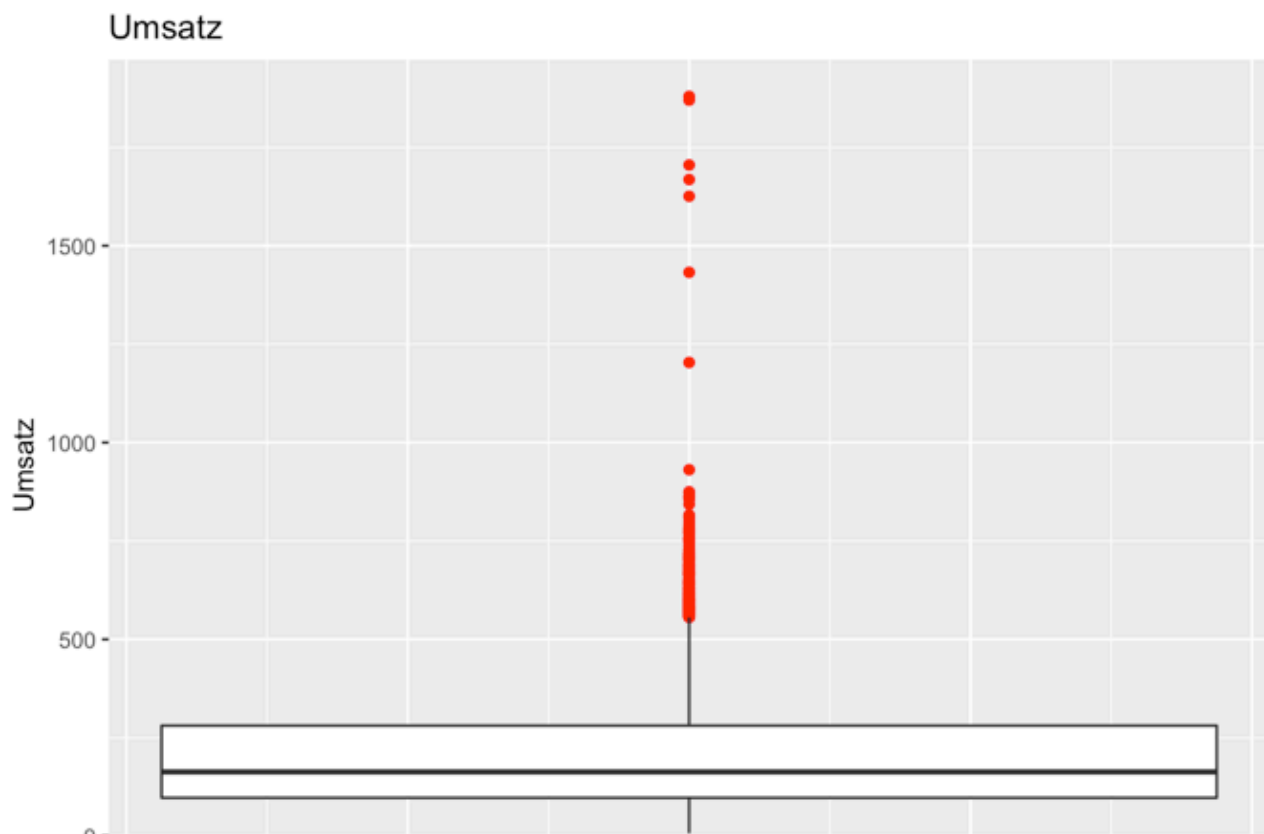
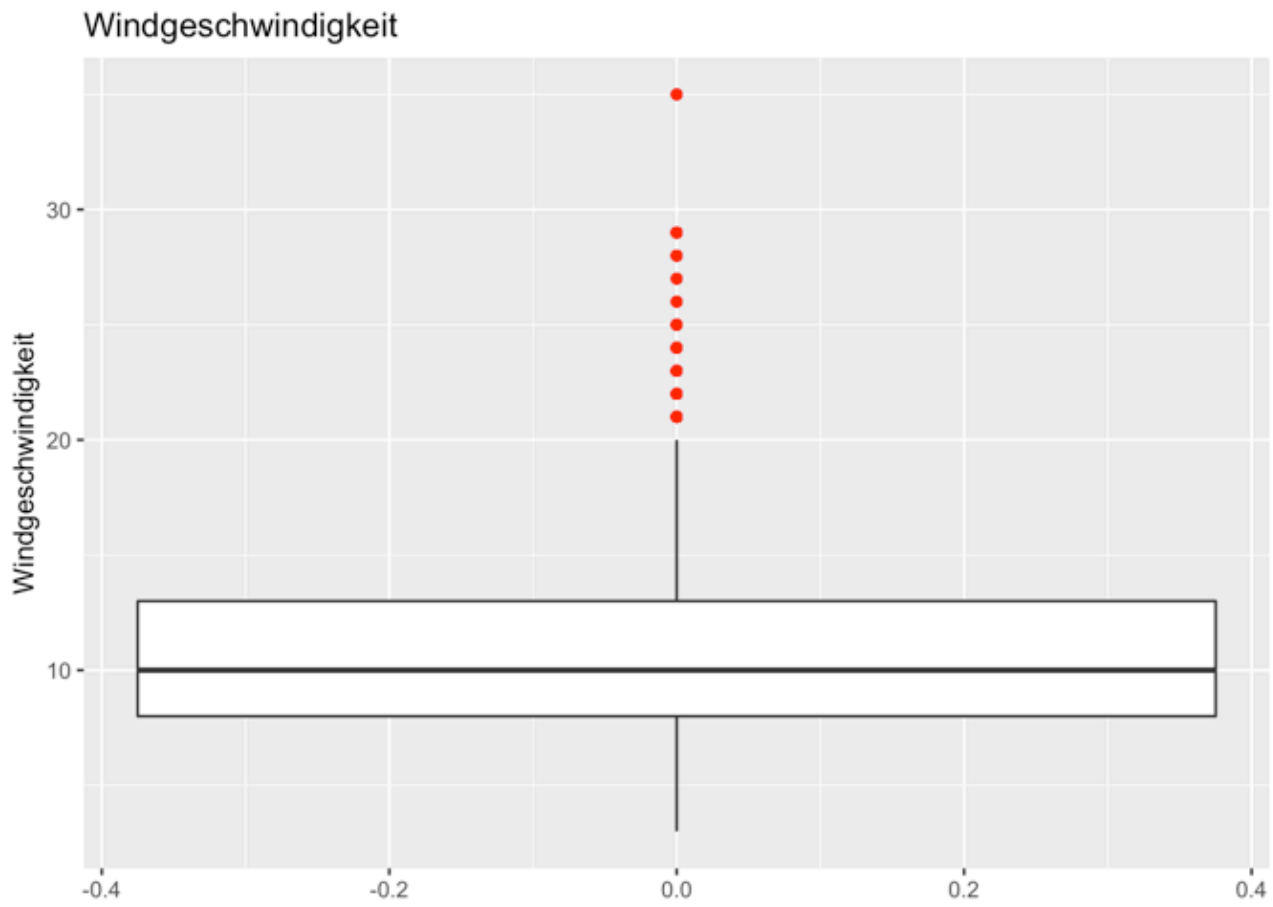
```
## Warning: Removed 81 rows containing non-finite values (stat_boxplot).
```

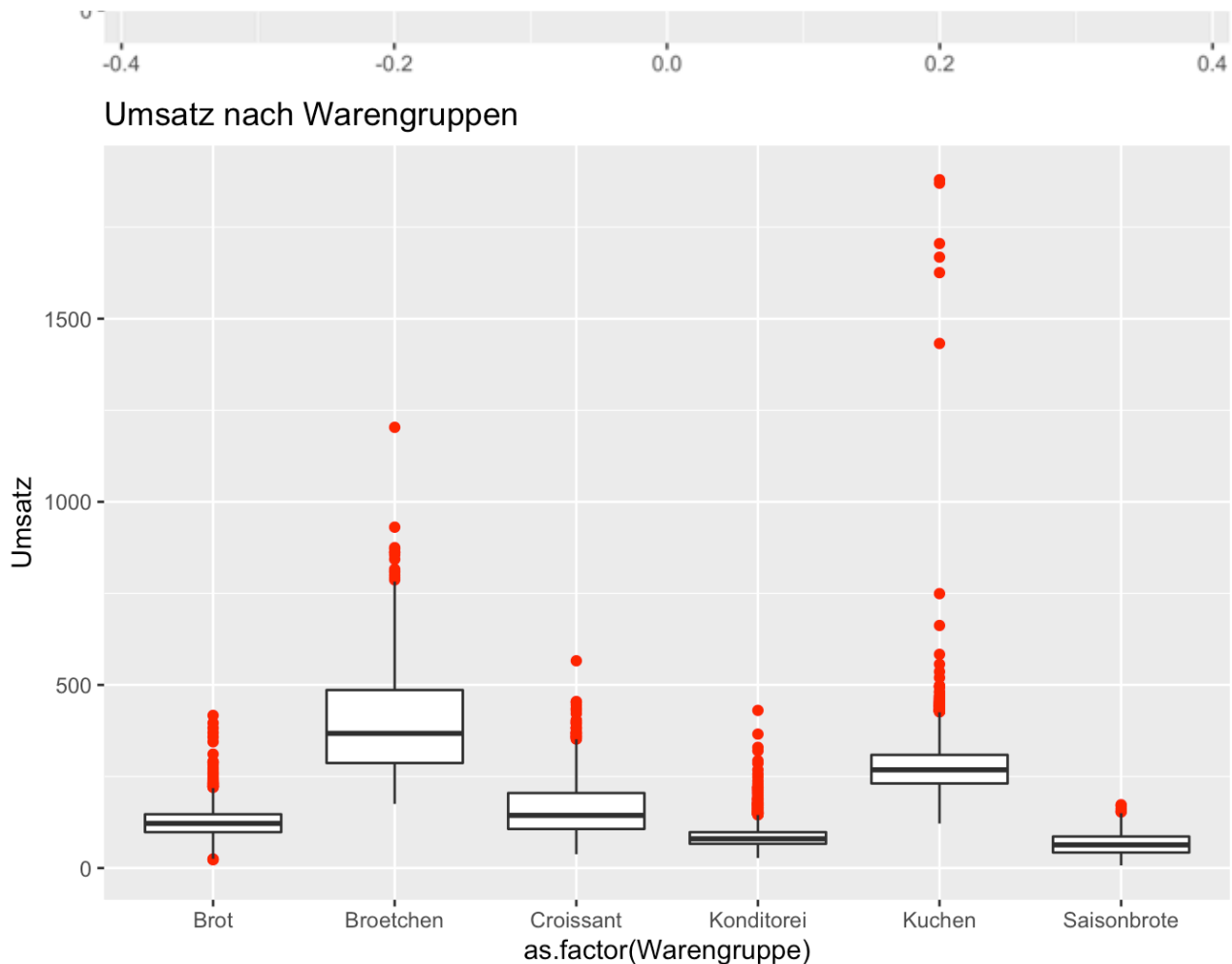


```
## Warning: Removed 135 rows containing non-finite values (stat_boxplot).
```



```
## Warning: Removed 81 rows containing non-finite values (stat_boxplot).
```





- Test auf Unterschiede der Mittelwerte im Umsatz per Warengruppen, um ggf. Warengruppen zusammenzufassen: nicht zu empfehlen, alle Unterschiede sig. von 0 verschieden

```
## Tukey multiple comparisons of means
## 95% family-wise confidence level
##
## Fit: aov(formula = Umsatz ~ Warengruppe, data = dataset)
##
## $Warengruppe
##
```

	diff	lwr	upr	p adj
Broetchen-Brot	274.47832	267.03128	281.92535	0.0000000
Croissant-Brot	40.11876	32.67173	47.56580	0.0000000
Konditorei-Brot	-36.80747	-44.30298	-29.31196	0.0000000
Kuchen-Brot	153.79543	146.34840	161.24246	0.0000000
Saisonbrote-Brot	-57.11407	-71.14025	-43.08790	0.0000000
Croissant-Broetchen	-234.35955	-241.80659	-226.91252	0.0000000
Konditorei-Broetchen	-311.28579	-318.78130	-303.79028	0.0000000
Kuchen-Broetchen	-120.68289	-128.12992	-113.23586	0.0000000
Saisonbrote-Broetchen	-331.59239	-345.61856	-317.56622	0.0000000
Konditorei-Croissant	-76.92624	-84.42175	-69.43072	0.0000000
Kuchen-Croissant	113.67666	106.22963	121.12370	0.0000000
Saisonbrote-Croissant	-97.23284	-111.25901	-83.20667	0.0000000
Kuchen-Konditorei	190.60290	183.10739	198.09841	0.0000000
Saisonbrote-Konditorei	-20.30660	-34.35858	-6.25463	0.0005478
Saisonbrote-Kuchen	-210.90950	-224.93568	-196.88333	0.0000000

Datenaufbereitung: Fehlende Werte imputieren

- KielerWoche und istFeiertage dichotom (0 wenn nicht zutreffend)
- Bewoelkung, Temperatur und Windgeschwindigkeit mittlere Werte imputieren

Data summary

Name	dataset
Number of rows	10899
Number of columns	11
Column type frequency:	
character	1
Date	1
factor	2
numeric	7
Group variables	
None	

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
Feiertage	0	1	1	25	0	6	0




Variable type: Date

skim_variable	n_missing	complete_rate	min	max	median	n_unique
Datum	0	1	2013-07-01	2019-06-06	2016-06-17	2121

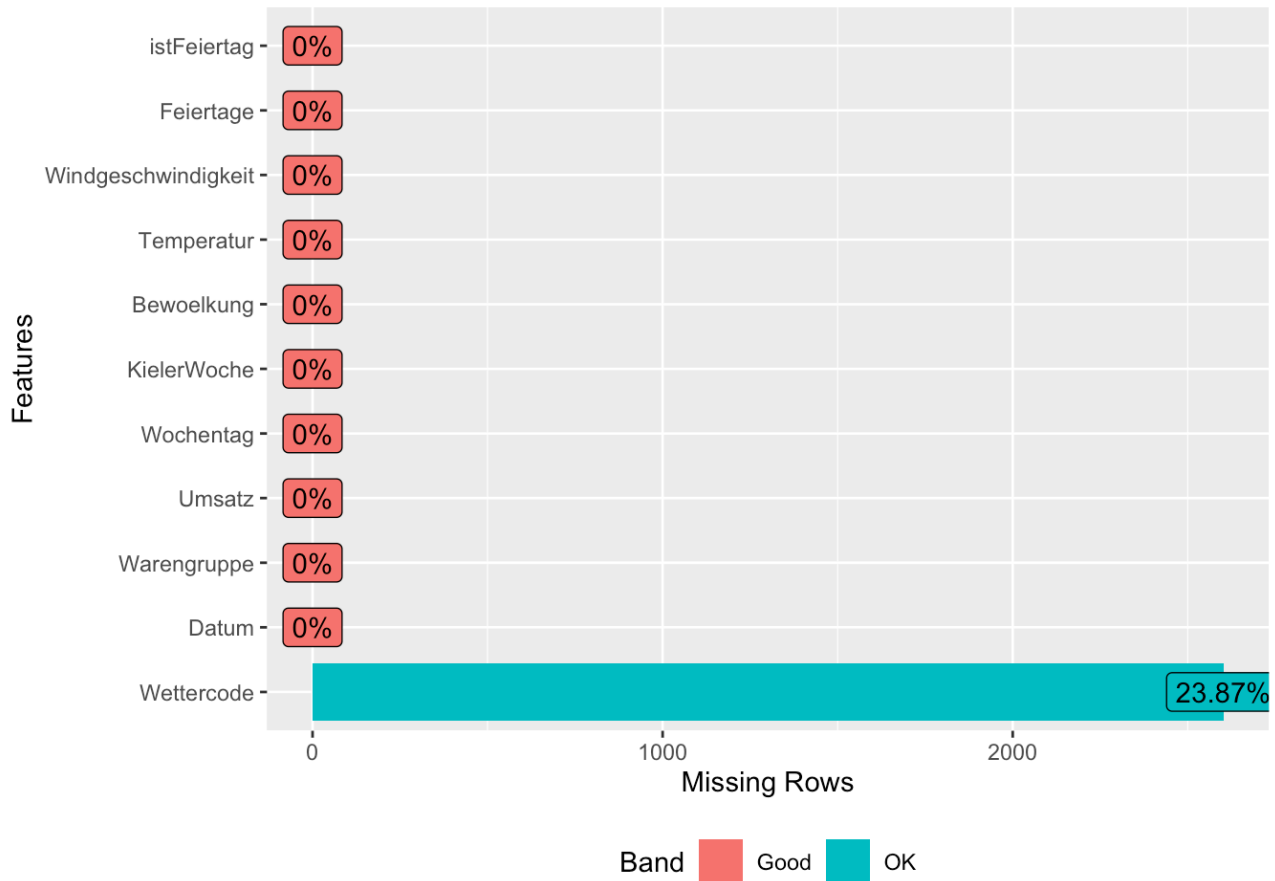
Variable type: factor

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
Warengruppe	0	1	FALSE	6	Bro: 2121, Bro: 2121, Cro: 2121, Kuc: 2121
Wochentag	0	1	FALSE	7	Son: 1570, Don: 1568, Sam: 1564, Die: 1562

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
Umsatz	0	1.00	206.66	142.81	7.05	97.53	163.30	280.81	1879.46	
KielerWoche	0	1.00	0.02	0.14	0.00	0.00	0.00	0.00	1.00	
Bewoelkung	0	1.00	4.73	2.63	0.00	3.00	6.00	7.00	8.00	

Temperatur	0	1.00	11.82	7.13	-8.47	6.12	11.38	17.62	32.67	
Windgeschwindigkeit	0	1.00	11.00	4.12	3.00	8.00	10.00	13.00	35.00	
Wettercode	2602	0.76	36.16	27.04	0.00	10.00	22.00	61.00	95.00	
istFeiertag	0	1.00	0.01	0.10	0.00	0.00	0.00	0.00	1.00	



Datenaufbereitung: Ausreißer bereinigen

```

#Ausreißer deckeln auf 3x Standardabweichung (Umsatz und Windgeschwindigkeit sind betroffen.)
#Bei 2x Standardabweichung werden 5% der Daten beeinflusst, das halte ich für zu viel. NUN 30
.05.2021
#Standardabweichung berechnen (sd(x))
sd_Umsatz<-sd(dataset$Umsatz, na.rm=TRUE)

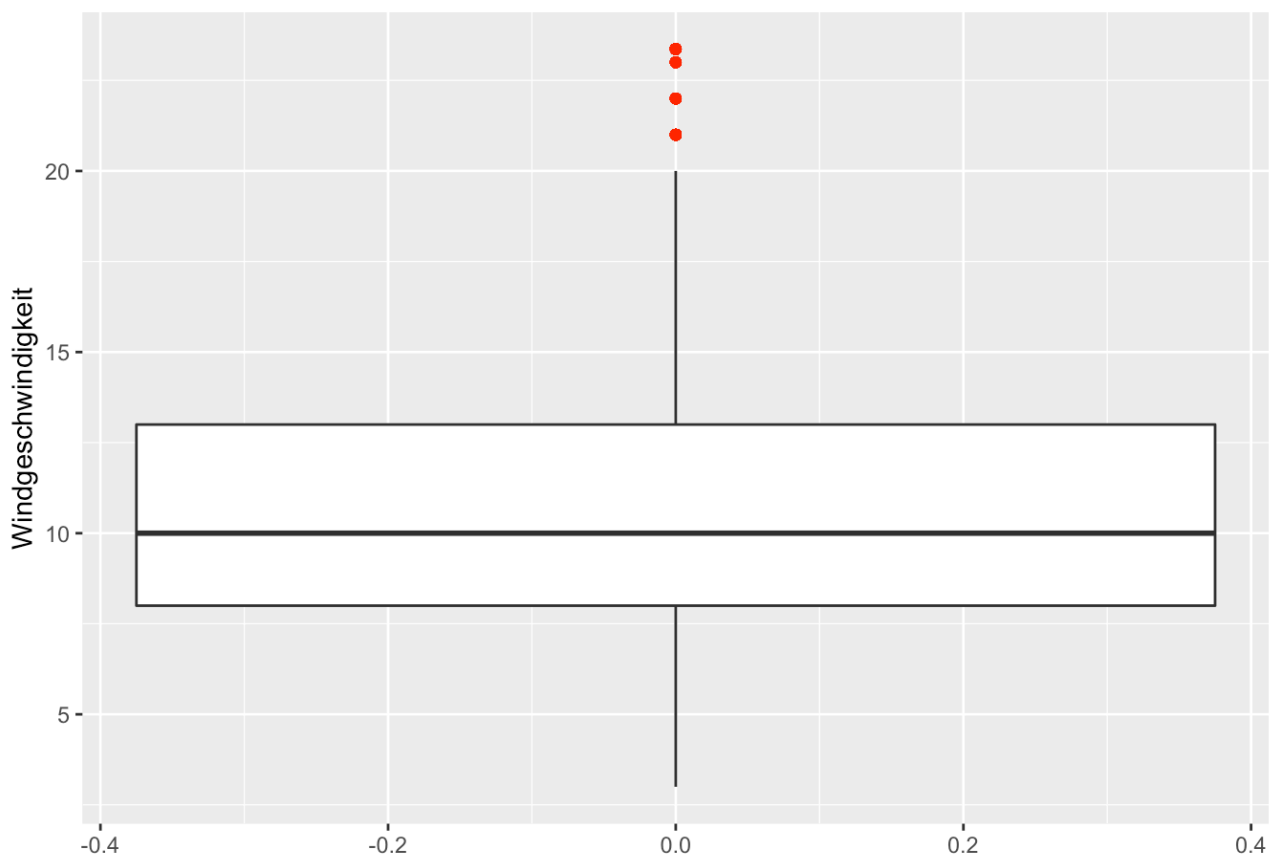
sd_Windgeschwindigkeit<-sd(dataset$Windgeschwindigkeit, na.rm=TRUE)

#Gefundene Werte auf 3x Standardabweichung setzen
dataset$Windgeschwindigkeit[dataset$Windgeschwindigkeit > (mean(dataset$Windgeschwindigkeit)
+ sd_Windgeschwindigkeit*3)]<- (mean(dataset$Windgeschwindigkeit) + sd_Windgeschwindigkeit*3)
dataset$Windgeschwindigkeit[dataset$Windgeschwindigkeit < (mean(dataset$Windgeschwindigkeit)
- sd_Windgeschwindigkeit*3)]<- (mean(dataset$Windgeschwindigkeit) - sd_Windgeschwindigkeit*3)
dataset$Umsatz[dataset$Umsatz > (mean(dataset$Umsatz) + sd_Umsatz*3)]<- (mean(dataset$Umsatz)
+ sd_Umsatz*3)
dataset$Umsatz[dataset$Umsatz < (mean(dataset$Umsatz) - sd_Umsatz*3)]<- (mean(dataset$Umsatz)
- sd_Umsatz*3)

ggplot(dataset) + geom_boxplot(aes(y=Windgeschwindigkeit),outlier.colour="red")+ coord_cartes
ian(ylim = c(min(dataset$Windgeschwindigkeit), max(dataset$Windgeschwindigkeit)))+ggtitle("Wi
ndgeschwindigkeit um Ausreißer (diff>3SD) bereinigt")

```

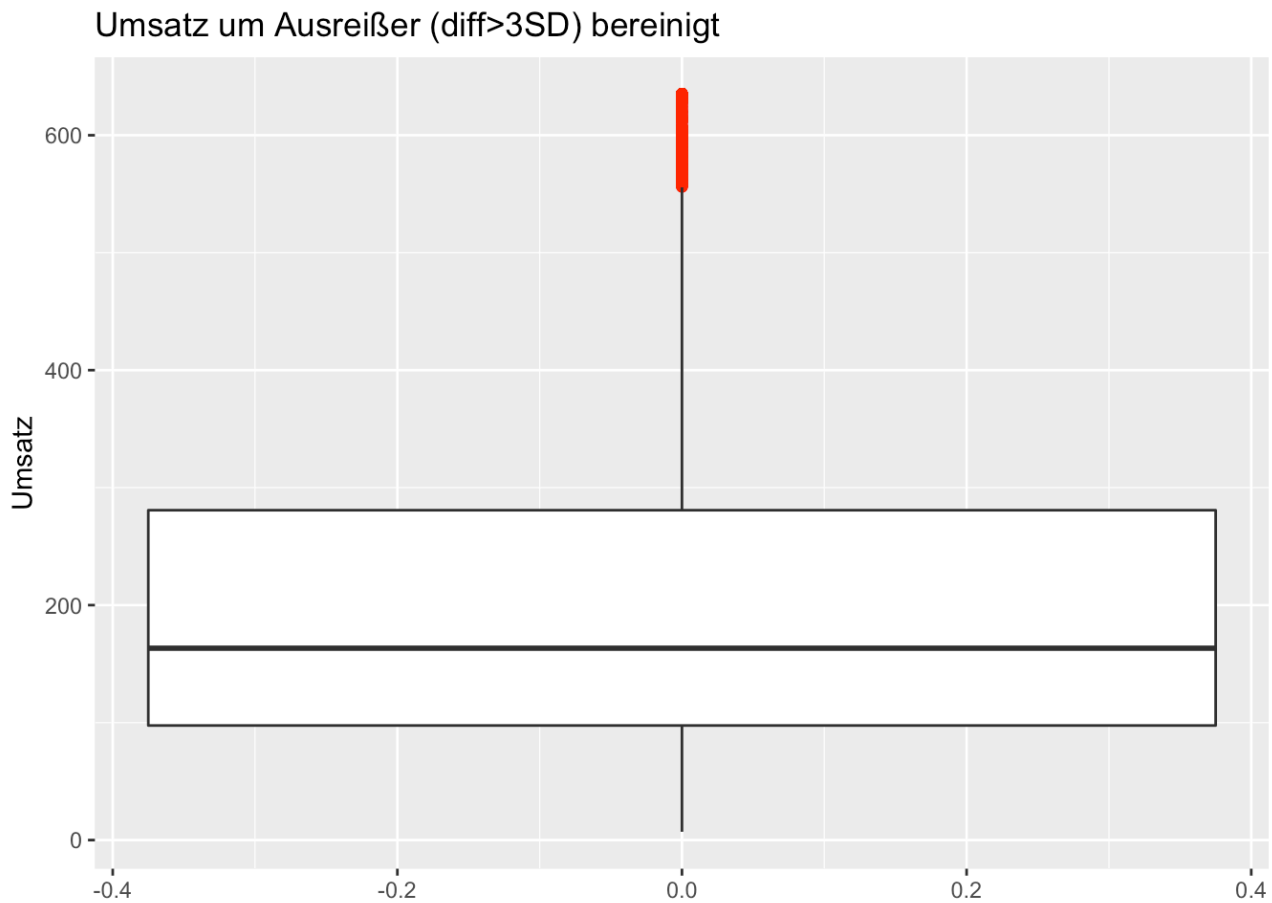
Windgeschwindigkeit um Ausreißer (diff>3SD) bereinigt



```

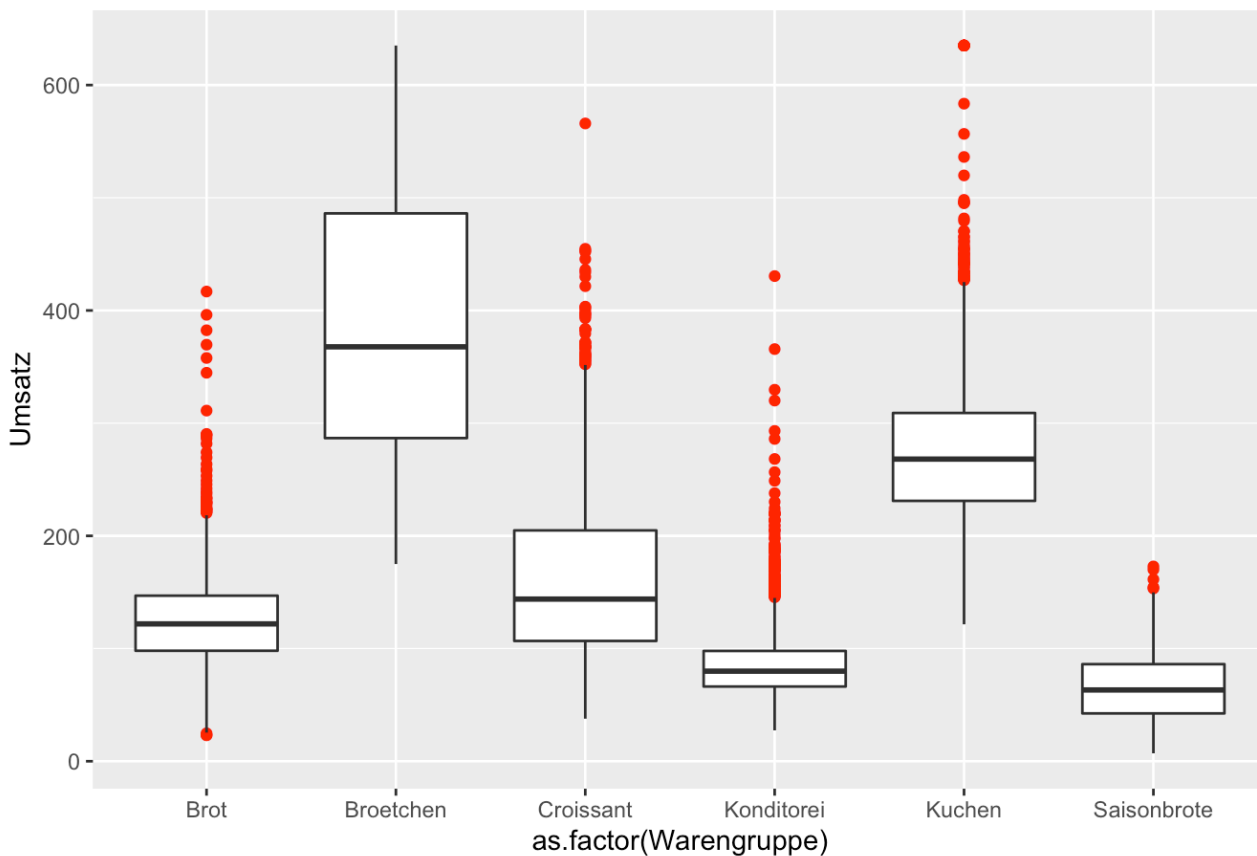
ggplot(dataset) + geom_boxplot(aes(y=Umsatz),outlier.colour="red")+ coord_cartesian(ylim = c(
min(dataset$Umsatz), max(dataset$Umsatz)))+ggtitle("Umsatz um Ausreißer (diff>3SD) bereinigt"
)

```



```
ggplot(dataset) +  
  geom_boxplot(aes(x=as.factor(Warengruppe), y=Umsatz), outlier.colour="red")+ coord_cartesian(y  
  lim = c(min(dataset$Umsatz), max(dataset$Umsatz)))+ggtitle("Umsatz nach Warengruppen um Ausre  
  ißer (diff>3SD) bereinigt")
```

Umsatz nach Warengruppen um Ausreißer (diff>3SD) bereinigt



```
rm(sd_Umsatz, sd_Windgeschwindigkeit)
```

Datensets für Modellierungen bereitstellen

Trainingsdatensätze erstellen

- data1 : Faktoren bei Warengruppe und Wochentag bleiben erhalten, Variablen Wettercode und (Bezeichnung der) Feiertage raus
- data2 : Wie data1, aber alle Variablen numerisch

```
str(data1)
```

```
## tibble [10,899 × 9] (S3: tbl_df/tbl/data.frame)
##  $ Datum           : Date[1:10899], format: "2013-07-01" "2013-07-02" ...
##  $ Warengruppe      : Factor w/ 6 levels "Brot","Broetchen",...: 1 1 1 1 1 1 1 1 1 1 ...
##  $ Umsatz           : num [1:10899] 149 160 112 169 171 ...
##  $ Wochentag         : Factor w/ 7 levels "Montag","Dienstag",...: 1 2 3 4 5 6 7 1 2 3 ...
##  $ KielerWoche       : num [1:10899] 0 0 0 0 0 0 0 0 0 0 ...
##  $ Bewoelkung        : num [1:10899] 6 3 7 7 5 0 0 0 0 2 ...
##  $ Temperatur        : num [1:10899] 17.8 17.3 21.1 18.9 20 ...
##  $ Windgeschwindigkeit: num [1:10899] 15 10 6 7 12 8 9 10 8 13 ...
##  $ istFeiertag       : num [1:10899] 0 0 0 0 0 0 0 0 0 0 ...
```

```
str(data2)
```

```
## tibble [10,899 × 9] (S3: tbl_df/tbl/data.frame)
## $ Datum           : Date[1:10899], format: "2013-07-01" "2013-07-02" ...
## $ Warengruppe     : num [1:10899] 1 1 1 1 1 1 1 1 1 1 ...
## $ Umsatz          : num [1:10899] 149 160 112 169 171 ...
## $ Wochentag        : num [1:10899] 1 2 3 4 5 6 7 1 2 3 ...
## $ KielerWoche      : num [1:10899] 0 0 0 0 0 0 0 0 0 0 ...
## $ Bewoelkung       : num [1:10899] 6 3 7 7 5 0 0 0 0 2 ...
## $ Temperatur       : num [1:10899] 17.8 17.3 21.1 18.9 20 ...
## $ Windgeschwindigkeit: num [1:10899] 15 10 6 7 12 8 9 10 8 13 ...
## $ istFeiertag      : num [1:10899] 0 0 0 0 0 0 0 0 0 0 ...
```

Vorhersagedatensatz mit gleicher Struktur erzeugen

Prädiktion wird vorbereitet für Umsätze 7.6.2019 (Freitag, erster Tag nach Ende der Umsatzdaten) und 09.6.2019 (Sonntag darauf) und zum Kontrast ein stürmischer kalter Mittwoch im Januar 09.01.2019

- pred_data1: Passend zu data1 mit Faktorenstrukturen
- pred_data2: Passend zu data2 mit nur numerischen Variablen

```
str(pred_data1)
```

```
## 'data.frame':    18 obs. of  8 variables:
## $ Warengruppe     : Factor w/ 6 levels "Brot","Broetchen",...: 1 1 1 2 2 2 3 3 3 4 ...
## $ Datum           : Date, format: "2019-06-07" "2019-06-09" ...
## $ Wochentag        : Factor w/ 7 levels "Montag","Dienstag",...: 5 7 3 5 7 3 5 7 3 5 ...
## $ KielerWoche      : num  0 0 0 0 0 0 0 0 0 0 ...
## $ Bewoelkung       : num  1 5 7 1 5 7 1 5 7 1 ...
## $ Temperatur       : num  17.45 18.65 3.14 17.45 18.65 ...
## $ Windgeschwindigkeit: num  10 10 22 10 10 22 10 10 22 10 ...
## $ istFeiertag      : num  0 0 0 0 0 0 0 0 0 0 ...
```

```
str(pred_data2)
```

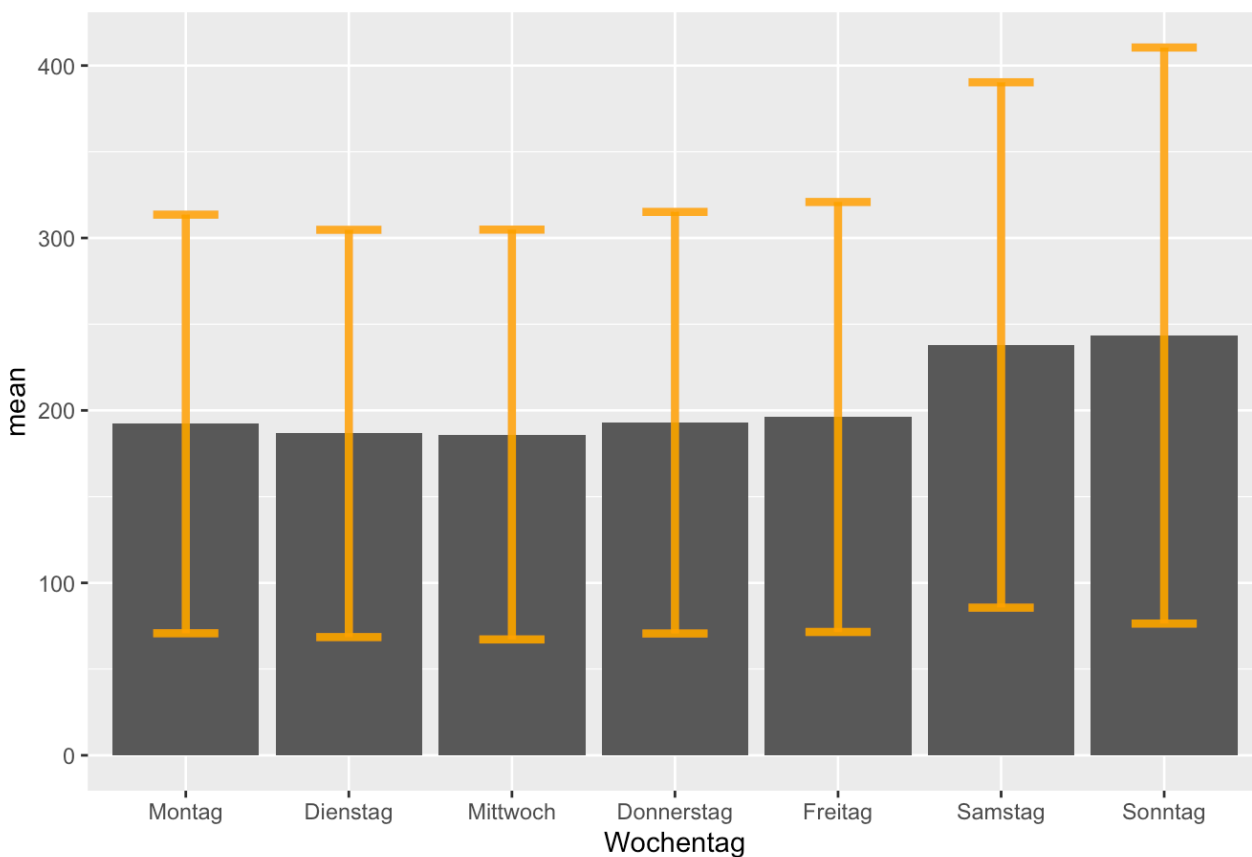
```
## 'data.frame':    18 obs. of  8 variables:
## $ Warengruppe     : num  1 1 1 2 2 2 3 3 3 4 ...
## $ Datum           : Date, format: "2019-06-07" "2019-06-09" ...
## $ Wochentag        : num  5 7 3 5 7 3 5 7 3 5 ...
## $ KielerWoche      : num  0 0 0 0 0 0 0 0 0 0 ...
## $ Bewoelkung       : num  1 5 7 1 5 7 1 5 7 1 ...
## $ Temperatur       : num  17.45 18.65 3.14 17.45 18.65 ...
## $ Windgeschwindigkeit: num  10 10 22 10 10 22 10 10 22 10 ...
## $ istFeiertag      : num  0 0 0 0 0 0 0 0 0 0 ...
```

Balken-Diagramme für selbsterstellte Variablen mit Angaben zur Streuung (Umsätze je Wochentag)

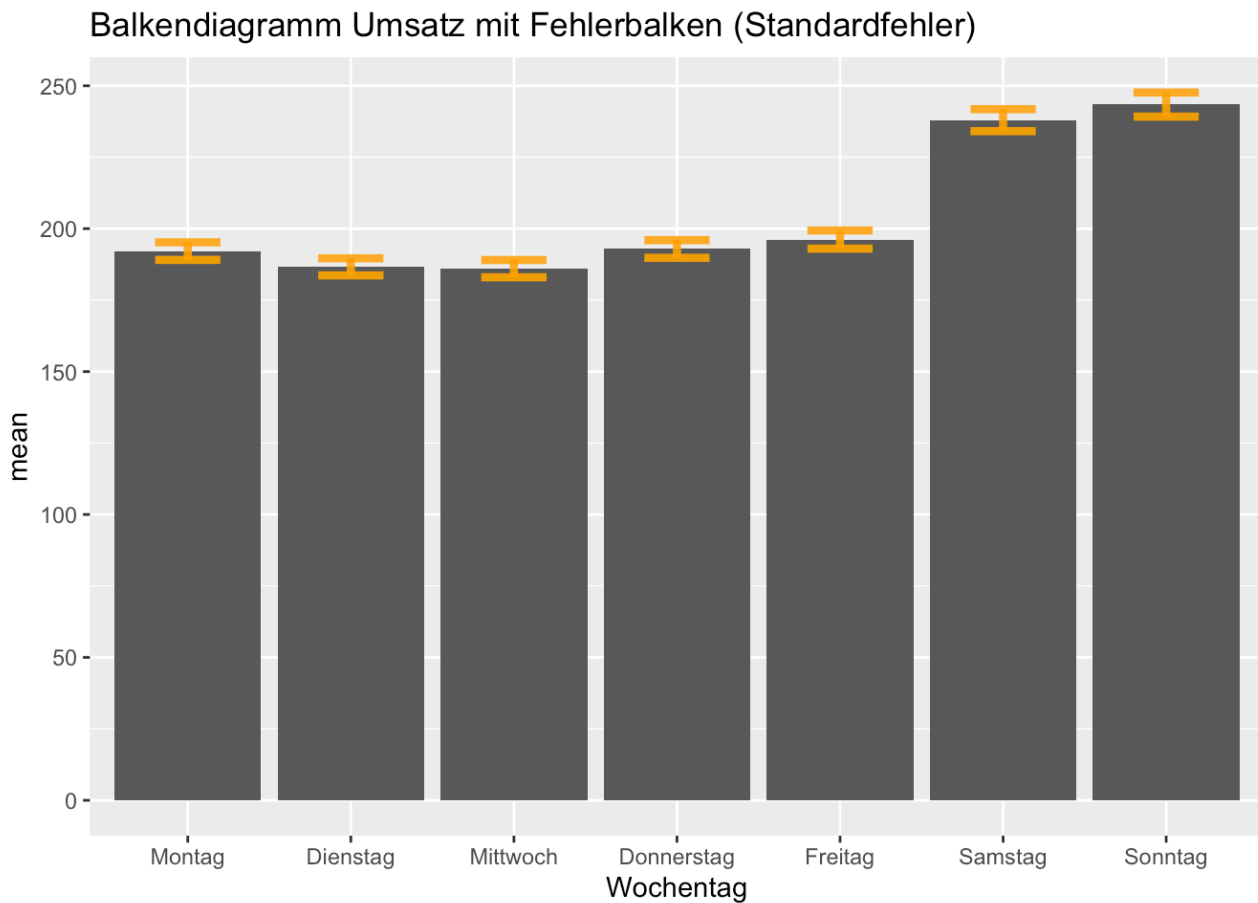
```
# Calculates mean, sd, se and IC für Umsatzdaten pro Wochentag
data1_sum <- data1 %>%
  group_by(Wochentag) %>%
  summarise(
    n=n(),
    mean=mean(Umsatz),
    sd=sd(Umsatz)
  ) %>%
  mutate( se=sd/sqrt(n)) %>%
  mutate( ic=se * qt((1-0.05)/2 + .5, n-1))

ggplot(data1_sum) +
  geom_bar( aes(x=Wochentag, y=mean), stat="identity") +
  geom_errorbar( aes(x=Wochentag, ymin=mean-sd, ymax=mean+sd), width=0.4, colour="orange", alpha=0.9, size=1.5) + scale_x_discrete(limits = ordentlicheWoche) +
  ggtitle("Balkendiagramm Umsatz mit Fehlerbalken (Standardabweichung) ")
```

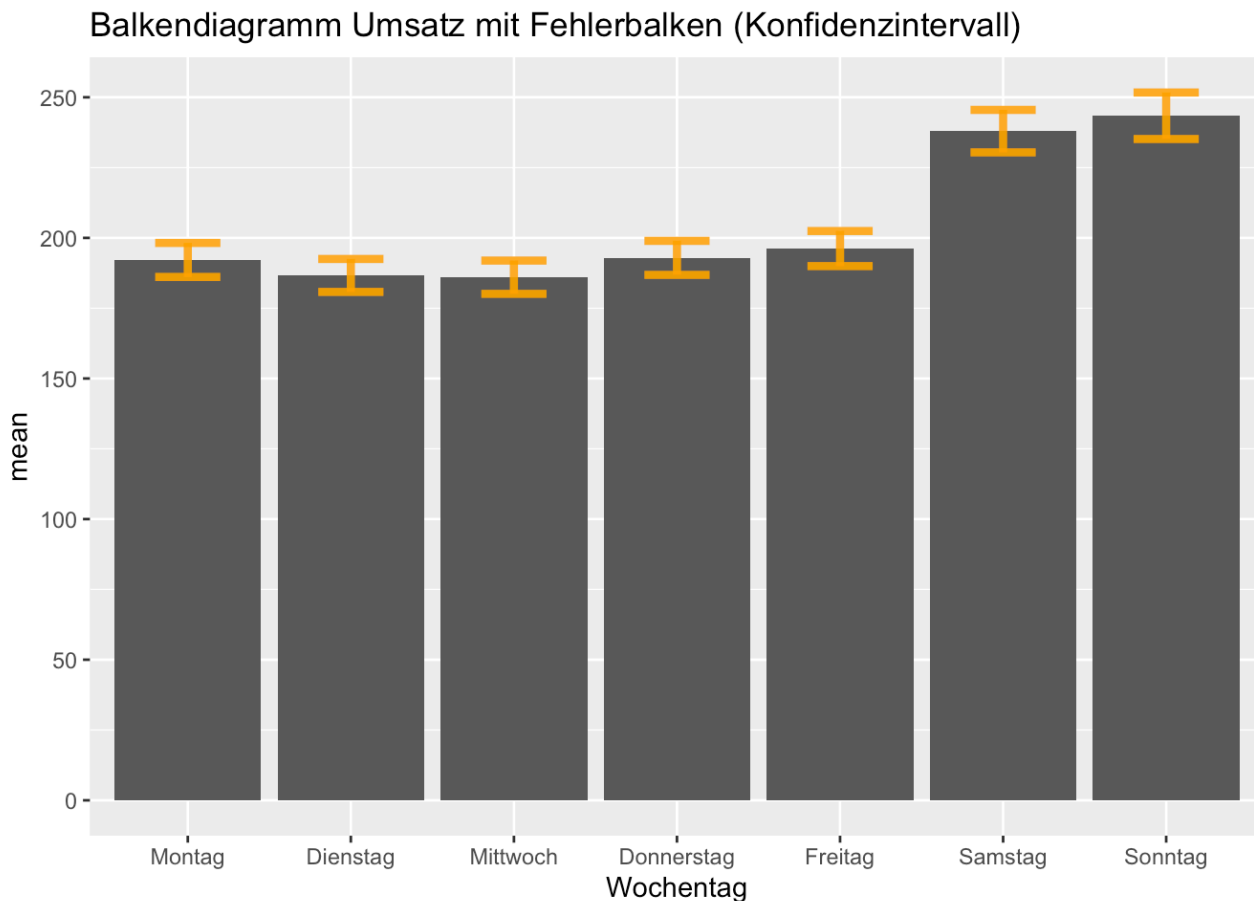
Balkendiagramm Umsatz mit Fehlerbalken (Standardabweichung)



```
ggplot(data1_sum) +
  geom_bar( aes(x=Wochentag, y=mean), stat="identity") +
  geom_errorbar( aes(x=Wochentag, ymin=mean-se, ymax=mean+se), width=0.4, colour="orange", alpha=0.9, size=1.5) + scale_x_discrete(limits = ordentlicheWoche) +
  ggtitle("Balkendiagramm Umsatz mit Fehlerbalken (Standardfehler) ")
```



```
ggplot(data1_sum) +
  geom_bar( aes(x=Wochentag, y=mean), stat="identity") +
  geom_errorbar( aes(x=Wochentag, ymin=mean-ic, ymax=mean+ic), width=0.4, colour="orange", al
pha=0.9, size=1.5) + scale_x_discrete(limits = ordentlicheWoche) +
  ggtitle("Balkendiagramm Umsatz mit Fehlerbalken (Konfidenzintervall) ")
```



Lineare Modelle/Supervised Learning

Optimierung des Modells mit schrittweiser Aufnahme von Variablen

	Model 1	Model 2	Model 3	Model 4	Model 5	Model 6	Model 7
(Intercept)	124.32 *** (1.62)	111.30 *** (2.28)	108.97 *** (2.28)	62.59 *** (2.34)	62.78 *** (2.34)	58.70 *** (2.90)	60.18 *** (3.29)
WarengruppeBroetchen	269.66 *** (2.30)	269.66 *** (2.18)	269.66 *** (2.18)	269.66 *** (2.00)	269.66 *** (2.00)	269.66 *** (2.00)	269.66 *** (2.00)
WarengruppeCroissant	40.12 *** (2.30)	40.12 *** (2.18)	40.12 *** (2.18)	40.12 *** (2.00)	40.12 *** (2.00)	40.12 *** (2.00)	40.12 *** (2.00)
WarengruppeKonditorei	-36.81 *** (2.31)	-36.97 *** (2.20)	-36.99 *** (2.19)	-36.83 *** (2.01)	-36.83 *** (2.01)	-36.82 *** (2.01)	-36.82 *** (2.01)
WarengruppeKuchen	150.73 ***	150.73 ***	150.73 ***	150.73 ***	150.73 ***	150.73 ***	150.73 ***

	(2.30)	(2.18)	(2.18)	(2.00)	(2.00)	(2.00)	(2.00)
WarengruppeSaisonbrote	-57.11 ***	-57.06 ***	-56.34 ***	-35.48 ***	-35.37 ***	-35.08 ***	-35.00 ***
	(4.32)	(4.11)	(4.10)	(3.80)	(3.80)	(3.80)	(3.80)
WochentagDienstag		-5.43 *	-3.33	-4.06	-4.04	-4.05	-4.01
		(2.55)	(2.55)	(2.34)	(2.34)	(2.34)	(2.34)
WochentagMittwoch		-6.71 **	-4.61	-5.78 *	-5.76 *	-5.79 *	-5.78 *
		(2.56)	(2.55)	(2.35)	(2.35)	(2.35)	(2.35)
WochentagDonnerstag		1.29	2.14	1.14	1.15	1.23	1.29
		(2.55)	(2.54)	(2.34)	(2.33)	(2.33)	(2.33)
WochentagFreitag		4.01	6.11 *	4.79 *	4.82 *	4.89 *	4.93 *
		(2.56)	(2.56)	(2.35)	(2.35)	(2.35)	(2.35)
WochentagSamstag		45.60 ***	47.71 ***	47.78 ***	47.57 ***	47.67 ***	47.71 ***
		(2.55)	(2.55)	(2.34)	(2.34)	(2.34)	(2.34)
WochentagSonntag		51.95 ***	54.05 ***	53.31 ***	53.09 ***	53.09 ***	53.08 ***
		(2.55)	(2.55)	(2.34)	(2.34)	(2.34)	(2.34)
istFeiertag			65.21 ***	58.20 ***	58.54 ***	58.18 ***	57.94 ***
			(6.69)	(6.16)	(6.15)	(6.15)	(6.16)
Temperatur				3.93 ***	3.89 ***	3.89 ***	3.86 ***
				(0.09)	(0.09)	(0.09)	(0.10)
KielerWoche					14.46 **	14.25 **	14.54 **
					(4.45)	(4.45)	(4.46)
Windgeschwindigkeit						0.37 *	0.38 *
						(0.16)	(0.16)
Bewoelkung							-0.25
							(0.26)
N	10899	10899	10899	10899	10899	10899	10899
R2	0.69	0.72	0.73	0.77	0.77	0.77	0.77

*** p < 0.001; ** p < 0.01; * p < 0.05.

```
## Analysis of Variance Table
##
## Model 1: Umsatz ~ Warengruppe
## Model 2: Umsatz ~ Warengruppe + Wochentag
## Model 3: Umsatz ~ Warengruppe + Wochentag + istFeiertag
## Model 4: Umsatz ~ Warengruppe + Wochentag + istFeiertag + Temperatur
## Model 5: Umsatz ~ Warengruppe + Wochentag + istFeiertag + Temperatur +
##           KielerWoche
## Model 6: Umsatz ~ Warengruppe + Wochentag + istFeiertag + Temperatur +
##           KielerWoche + Windgeschwindigkeit
## Model 7: Umsatz ~ Warengruppe + Wochentag + istFeiertag + Temperatur +
##           KielerWoche + Windgeschwindigkeit + Bewoelkung
##   Res.Df      RSS Df Sum of Sq      F    Pr(>F)
## 1  10893 60872235
## 2  10887 55094668  6   5777567  227.0783 < 2.2e-16 ***
## 3  10886 54617735  1    476933  112.4707 < 2.2e-16 ***
## 4  10885 46217848  1   8399887 1980.8675 < 2.2e-16 ***
## 5  10884 46173133  1    44716   10.5449  0.001169 **
## 6  10883 46149114  1    24019    5.6642  0.017332 *
## 7  10882 46145221  1     3892    0.9179  0.338053
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Entscheidung für das Modell 6 **Umsatz ~ Warengruppe + Wochentag + istFeiertag + Temperatur + KielerWoche + Windgeschwindigkeit** mit 0,715 Varianzaufklärung

```
## MODEL INFO:
## Observations: 10899
## Dependent Variable: Umsatz
## Type: OLS linear regression
##
## MODEL FIT:
## F(15,10883) = 2410.29, p = 0.00
## R2 = 0.77
## Adj. R2 = 0.77
##
## Standard errors: OLS
## -----
##               Est.    S.E.    t val.    p
## -----
## (Intercept)      58.70    2.90    20.21    0.00
## WarengruppeBroetchen 269.66    2.00   134.85    0.00
## WarengruppeCroissant  40.12    2.00    20.06    0.00
## WarengruppeKonditorei -36.82    2.01   -18.29    0.00
## WarengruppeKuchen   150.73    2.00    75.38    0.00
## WarengruppeSaisonbrote -35.08    3.80    -9.24    0.00
## WochentagDienstag    -4.05    2.34    -1.73    0.08
## WochentagMittwoch    -5.79    2.35    -2.46    0.01
## WochentagDonnerstag    1.23    2.33     0.53    0.60
## WochentagFreitag     4.89    2.35     2.08    0.04
## WochentagSamstag     47.67    2.34    20.34    0.00
## WochentagSonntag     53.09    2.34    22.68    0.00
## istFeiertag         58.18    6.15     9.45    0.00
## Temperatur          3.89    0.09    43.69    0.00
## KielerWoche         14.25    4.45     3.20    0.00
## Windgeschwindigkeit  0.37    0.16     2.38    0.02
## -----
```

Prädiktion 1 (Lineare Modelle/Supervised Learning)

aka: Ich nehme ein lineares Modell, das auf einen Datensatz gefittet wurde und sage mit den dort geschätzten Koeffizienten die Zielvariable auf Basis von einem neuen Datensatz vorher

```

#Vorhersage berechnen
pred_lm <- predict(object=LMfinal, newdata = pred_data1, type = "response", interval = "confidence", na.action = na.exclude)

#print(pred_lm)
#class(pred_lm)#is ne doofe Matrix--> für Nutzung von dplyr-Routinen as.data.frame()

pred_lm <- dplyr::bind_cols(pred_data1, as.data.frame(pred_lm)) #in dieser Datei sind "fit" die Umsatz-Prädiktionen für die jeweiligen Datenpunkte, lwr und upr sind die Grenzen der Konfidenzintervalle

pred_lm<- rename(pred_lm,Prediction_Umsatz_LM = fit,UntereGrenzeKonfInt=lwr,ObereGrenzeKonfInt=upr)

anzeige <-dplyr::filter(pred_lm,Datum==ymd(20190607))
anzeige <-dplyr::select(anzeige, c("Warengruppe","Prediction_Umsatz_LM","UntereGrenzeKonfInt","ObereGrenzeKonfInt"))
pred_lm

```

##	Warengruppe	Datum	Wochentag	KielerWoche	Bewoelkung	Temperatur
## 1	Brot	2019-06-07	Freitag	0	1	17.4500
## 2	Brot	2019-06-09	Sonntag	0	5	18.6500
## 3	Brot	2019-01-09	Mittwoch	0	7	3.1375
## 4	Broetchen	2019-06-07	Freitag	0	1	17.4500
## 5	Broetchen	2019-06-09	Sonntag	0	5	18.6500
## 6	Broetchen	2019-01-09	Mittwoch	0	7	3.1375
## 7	Croissant	2019-06-07	Freitag	0	1	17.4500
## 8	Croissant	2019-06-09	Sonntag	0	5	18.6500
## 9	Croissant	2019-01-09	Mittwoch	0	7	3.1375
## 10	Konditorei	2019-06-07	Freitag	0	1	17.4500
## 11	Konditorei	2019-06-09	Sonntag	0	5	18.6500
## 12	Konditorei	2019-01-09	Mittwoch	0	7	3.1375
## 13	Kuchen	2019-06-07	Freitag	0	1	17.4500
## 14	Kuchen	2019-06-09	Sonntag	0	5	18.6500
## 15	Kuchen	2019-01-09	Mittwoch	0	7	3.1375
## 16	Saisonbrote	2019-06-07	Freitag	0	1	17.4500
## 17	Saisonbrote	2019-06-09	Sonntag	0	5	18.6500
## 18	Saisonbrote	2019-01-09	Mittwoch	0	7	3.1375
##	Windgeschwindigkeit	istFeiertag	Prediction_Umsatz_LM	UntereGrenzeKonfInt		
## 1	10	0	135.20687	130.98910		
## 2	10	0	188.08222	183.81511		
## 3	22	0	73.24846	67.77004		
## 4	10	0	404.86622	400.64845		
## 5	10	0	457.74157	453.47446		
## 6	22	0	342.90781	337.42939		
## 7	10	0	175.32563	171.10787		
## 8	10	0	228.20098	223.93387		
## 9	22	0	113.36722	107.88880		
## 10	10	0	98.38697	94.14281		
## 11	10	0	151.26232	146.97850		
## 12	22	0	36.42856	30.92545		
## 13	10	0	285.93224	281.71447		
## 14	10	0	338.80759	334.54048		
## 15	22	0	223.97383	218.49541		

## 16	10	0	100.12620	92.42066
## 17	10	0	153.00155	145.24894
## 18	22	0	38.16779	29.86332
##	ObereGrenzeKonfInt			
## 1	139.42464			
## 2	192.34933			
## 3	78.72688			
## 4	409.08399			
## 5	462.00868			
## 6	348.38623			
## 7	179.54340			
## 8	232.46809			
## 9	118.84564			
## 10	102.63113			
## 11	155.54615			
## 12	41.93167			
## 13	290.15001			
## 14	343.07470			
## 15	229.45225			
## 16	107.83173			
## 17	160.75415			
## 18	46.47225			

Die Vorhersage für den ersten Tag (2019-06-07) nach Ende der Datenreihe ist:

##	Warengruppe	Prediction_Umsatz_LM	UntereGrenzeKonfInt	ObereGrenzeKonfInt
## 1	Brot	135.20687	130.98910	139.4246
## 4	Broetchen	404.86622	400.64845	409.0840
## 7	Croissant	175.32563	171.10787	179.5434
## 10	Konditorei	98.38697	94.14281	102.6311
## 13	Kuchen	285.93224	281.71447	290.1500
## 16	Saisonbrote	100.12620	92.42066	107.8317

Support Vector Maschinen

Trainings- und Testdatensätze erstellen

Achtung: Im Moment auf 10% Trainingsdaten, wegen Performanz! Am Ende wieder anpassen

```
data<-data2

# Zufallszähler setzen (um die zufällige Partitionierung bei jedem Durchlauf gleich zu halten
)
set.seed(1)

#Datensatzzeilen shuffeln
new_row_order<-sample(nrow(data))
data<-data[new_row_order,]

# Zufällige Ziehung von Indizes für die Zeilen des Datensatzes, die dem Trainingsdatensatz zug
eordnet werden, Umfang: 80% (im Moment 10%)
indices_train <- sample(seq_len(nrow(data)), size = floor(0.10 * nrow(data)))

# Definition des Trainings- und Testdatensatz durch Selektion bzw. Deselektion der entspreche
nden Datenzeilen
train_data <- data[indices_train, ]
test_data <- data[-indices_train, ]
```

SVM Trainieren (ohne Tuning)

```
model_svm <- svm(Umsatz ~ Warengruppe+Wochentag, train_data)
summary(model_svm)
```

```
##
## Call:
## svm(formula = Umsatz ~ Warengruppe + Wochentag, data = train_data)
##
##
## Parameters:
##   SVM-Type:  eps-regression
##   SVM-Kernel: radial
##         cost:  1
##        gamma: 0.5
##     epsilon: 0.1
##
##
## Number of Support Vectors:  806
```

Kreuzvalidierung

```
test_data_svm <-predict(model_svm,test_data,na.action = na.pass)
mape(test_data$Umsatz,test_data_svm) #Mean Absolute Percent Error,
```

```
## [1] 0.2774538
```

```
#rse: sum(error^2)/sum(actual-mean(actual))  
#r^2=1-rse #laut Steffen Brandt  
  
r_squared_svm<-1-rse(test_data$Umsatz,test_data_svm)  
r_squared_svm
```

```
## [1] 0.7266911
```

##Prädiktion 2 (SVM ohne Tuning)

```
pred_svm <- predict(model_svm, pred_data2, na.action = na.pass)# Daten Vorhersagen  
  
pred_svm <- dplyr::bind_cols(pred_data2, as.data.frame(pred_svm )) #zusammenfügen  
  
pred_svm<- rename(pred_svm,Prediction_Umsatz_SVM = pred_svm)#Variablenbenennung anpassen  
  
#Faktoren wieder schick machen  
pred_svm$Wochentag <- factor(weekdays(pred_svm$Datum),  
levels=ordentlicheWoche)  
pred_svm$Warengruppe <- factor(pred_svm$Warengruppe,  
levels = c(1,2,3,4,5,6),  
labels = c("Brot", "Broetchen", "Croissant", "Konditorei", "Kuchen", "Saisonbrote"))  
  
anzeige <-dplyr::filter(pred_svm,Datum==ymd(20190607))  
anzeige <-dplyr::select(anzeige, c("Warengruppe", "Prediction_Umsatz_SVM"))  
  
pred_svm
```

##	Warengruppe	Datum	Wochentag	KielerWoche	Bewoelkung	Temperatur
## 1	Brot	2019-06-07	Freitag	0	1	17.4500
## 2	Brot	2019-06-09	Sonntag	0	5	18.6500
## 3	Brot	2019-01-09	Mittwoch	0	7	3.1375
## 4	Broetchen	2019-06-07	Freitag	0	1	17.4500
## 5	Broetchen	2019-06-09	Sonntag	0	5	18.6500
## 6	Broetchen	2019-01-09	Mittwoch	0	7	3.1375
## 7	Croissant	2019-06-07	Freitag	0	1	17.4500
## 8	Croissant	2019-06-09	Sonntag	0	5	18.6500
## 9	Croissant	2019-01-09	Mittwoch	0	7	3.1375
## 10	Konditorei	2019-06-07	Freitag	0	1	17.4500
## 11	Konditorei	2019-06-09	Sonntag	0	5	18.6500
## 12	Konditorei	2019-01-09	Mittwoch	0	7	3.1375
## 13	Kuchen	2019-06-07	Freitag	0	1	17.4500
## 14	Kuchen	2019-06-09	Sonntag	0	5	18.6500
## 15	Kuchen	2019-01-09	Mittwoch	0	7	3.1375
## 16	Saisonbrote	2019-06-07	Freitag	0	1	17.4500
## 17	Saisonbrote	2019-06-09	Sonntag	0	5	18.6500
## 18	Saisonbrote	2019-01-09	Mittwoch	0	7	3.1375
##	Windgeschwindigkeit	istFeiertag	Prediction_Umsatz_SVM			
## 1	10	0	139.29890			
## 2	10	0	102.46227			
## 3	22	0	119.77695			
## 4	10	0	373.23100			
## 5	10	0	444.58441			
## 6	22	0	323.16086			
## 7	10	0	146.96562			
## 8	10	0	246.05492			
## 9	22	0	124.89878			
## 10	10	0	76.52544			
## 11	10	0	124.46991			
## 12	22	0	75.97449			
## 13	10	0	261.45276			
## 14	10	0	276.18899			
## 15	22	0	253.99799			
## 16	10	0	93.58963			
## 17	10	0	157.11996			
## 18	22	0	76.49333			

Die Vorhersage für den ersten Tag (2019-06-07) nach Ende der Datenreihe ist:

##	Warengruppe	Prediction_Umsatz_SVM
## 1	Brot	139.29890
## 4	Broetchen	373.23100
## 7	Croissant	146.96562
## 10	Konditorei	76.52544
## 13	Kuchen	261.45276
## 16	Saisonbrote	93.58963

SVM Trainieren (mit Tuning)


```
svm_tune <- tune(svm, Umsatz ~ Warengruppe + Wochentag, data=train_data, ranges = list(epsilon
= seq(0.2,1,0.1), cost = 2^(2:3)))
summary(svm_tune)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##   epsilon cost
##     0.3      8
##
## - best performance: 4550.13
##
## - Detailed performance results:
##   epsilon cost   error dispersion
## 1      0.2      4 4625.116    1500.636
## 2      0.3      4 4600.966    1469.471
## 3      0.4      4 4705.820    1487.649
## 4      0.5      4 4841.237    1458.590
## 5      0.6      4 5066.137    1361.817
## 6      0.7      4 5449.668    1276.171
## 7      0.8      4 5889.816    1202.699
## 8      0.9      4 6674.801    1209.729
## 9      1.0      4 7622.299    1107.956
## 10     0.2      8 4563.291    1489.614
## 11     0.3      8 4550.130    1467.544
## 12     0.4      8 4629.443    1470.618
## 13     0.5      8 4757.600    1440.336
## 14     0.6      8 4943.481    1335.859
## 15     0.7      8 5346.577    1231.085
## 16     0.8      8 5803.544    1170.991
## 17     0.9      8 6370.311    1115.436
## 18     1.0      8 7304.838    1077.447
```

Kreuzvalidierung

```
test_data_svm<-predict(svm_tune$best.model,test_data,na.action = na.pass)
mape(test_data$Umsatz,test_data_svm) #Mean Absolute Percent Error
```

```
## [1] 0.2770731
```

```
r_squared_svmtune<-1-rse(test_data$Umsatz,test_data_svm)
r_squared_svmtune
```

```
## [1] 0.7448893
```

Prädiktion 3 (SVM mit Tuning)

```
pred_svm_tune <- predict(svm_tune$best.model, pred_data2, na.action = na.pass) # Daten Vorhersagen

pred_svm_tune <- dplyr::bind_cols(pred_data2, as.data.frame(pred_svm_tune)) # zusammenfügen

pred_svm_tune <- rename(pred_svm_tune, Prediction_Umsatz_SVM_tune = pred_svm_tune) # Variablenbenennung anpassen

# Faktoren wieder schick machen
pred_svm_tune$Wochentag <- factor(weekdays(pred_svm_tune$Datum),
  levels=ordentlicheWoche)
pred_svm_tune$Warengruppe <- factor(pred_svm_tune$Warengruppe,
  levels = c(1,2,3,4,5,6),
  labels = c("Brot", "Broetchen", "Croissant", "Konditorei", "Kuchen", "Saisonbrote"))

anzeige <- dplyr::filter(pred_svm_tune, Datum == ymd(20190607))
anzeige <- dplyr::select(anzeige, c("Warengruppe", "Prediction_Umsatz_SVM_tune"))

pred_svm_tune
```

```
##      Warengruppe      Datum Wochentag KielerWoche Bewoelkung Temperatur
## 1      Brot 2019-06-07   Freitag          0          1    17.4500
## 2      Brot 2019-06-09   Sonntag          0          5    18.6500
## 3      Brot 2019-01-09  Mittwoch          0          7     3.1375
## 4    Broetchen 2019-06-07   Freitag          0          1    17.4500
## 5    Broetchen 2019-06-09   Sonntag          0          5    18.6500
## 6    Broetchen 2019-01-09  Mittwoch          0          7     3.1375
## 7    Croissant 2019-06-07   Freitag          0          1    17.4500
## 8    Croissant 2019-06-09   Sonntag          0          5    18.6500
## 9    Croissant 2019-01-09  Mittwoch          0          7     3.1375
## 10 Konditorei 2019-06-07   Freitag          0          1    17.4500
## 11 Konditorei 2019-06-09   Sonntag          0          5    18.6500
## 12 Konditorei 2019-01-09  Mittwoch          0          7     3.1375
## 13      Kuchen 2019-06-07   Freitag          0          1    17.4500
## 14      Kuchen 2019-06-09   Sonntag          0          5    18.6500
## 15      Kuchen 2019-01-09  Mittwoch          0          7     3.1375
## 16 Saisonbrote 2019-06-07   Freitag          0          1    17.4500
## 17 Saisonbrote 2019-06-09   Sonntag          0          5    18.6500
## 18 Saisonbrote 2019-01-09  Mittwoch          0          7     3.1375
##      Windgeschwindigkeit istFeiertag Prediction_Umsatz_SVMtune
## 1              10          0          143.17491
## 2              10          0          100.03611
## 3              22          0          121.13788
## 4              10          0          376.98689
## 5              10          0          488.99056
## 6              22          0          344.50752
## 7              10          0          143.61208
## 8              10          0          217.82495
## 9              22          0          139.26023
## 10             10          0           84.67616
## 11             10          0          121.64906
## 12             22          0           83.53556
## 13             10          0          270.14418
## 14             10          0          303.90408
## 15             22          0          260.70306
## 16             10          0           89.08484
## 17             10          0          107.44911
## 18             22          0           70.82903
```

Die Vorhersage für den ersten Tag (2019-06-07) nach Ende der Datenreihe ist:

```
##      Warengruppe Prediction_Umsatz_SVMtune
## 1      Brot          143.17491
## 4    Broetchen          376.98689
## 7    Croissant          143.61208
## 10 Konditorei           84.67616
## 13      Kuchen          270.14418
## 16 Saisonbrote           89.08484
```

```
## Warning in rm(anzeige, indices_train, new_row_order, test_data_svm): Objekt
## 'indices_train' nicht gefunden
```

```
## Warning in rm(anzeige, indices_train, new_row_order, test_data_svm): Objekt
## 'new_row_order' nicht gefunden
```

Neuronale Netze

Vorbereitung

Installation von Python und TensorFlow (nur einmalig nötig, im Hintergrund)

Benötigte Pakete laden (im Hintergrund)

Daten aufbereiten (im Hintergrund)

Training des Neuronalen Netzes

Definition des Neuronalen Netzes

```
# Import needed Python libraries and functions
import numpy as np
import tensorflow as tf
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Dropout
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.layers.experimental import preprocessing
# Create a Normalization layer and use the means and variances of the training features for the normalization
normalizer = preprocessing.Normalization()
normalizer.adapt(r.training_features.values)
# The argument "shape" for the definition of the input layer must include the number of variables (features) used for the model. To automatically calculate this number, we use the "r.training_features.keys()", which returns the list of variable names of the dataframe "training_features". Further, the function len() returns the length of this list of variable names (i.e. the number of variables in the input).
inputs = tf.keras.Input(shape=[len(r.training_features.keys())])
# Normalization layer
x = normalizer(inputs)
# 1st hidden layer
x = Dense(10, activation='relu')(x)
x = Dropout(.2)(x)
# 2nd hidden layer
x = Dense(4, activation='relu')(x)
# Output layer
output = tf.keras.layers.Dense(1)(x)
# Model definition
model = tf.keras.Model(inputs, output)
# Ausgabe einer Zusammenfassung zur Form des Modells, das geschätzt wird (nicht notwendig)
model.summary()
```

```
## Model: "model"
##
## _____
## Layer (type)                Output Shape                Param #
## =====
## input_1 (InputLayer)        [(None, 15)]                0
## _____
## normalization (Normalization (None, 15))                31
## _____
## dense (Dense)                (None, 10)                160
## _____
## dropout (Dropout)            (None, 10)                0
## _____
## dense_1 (Dense)              (None, 4)                 44
## _____
## dense_2 (Dense)              (None, 1)                 5
## =====
## Total params: 240
## Trainable params: 209
## Non-trainable params: 31
## _____
```

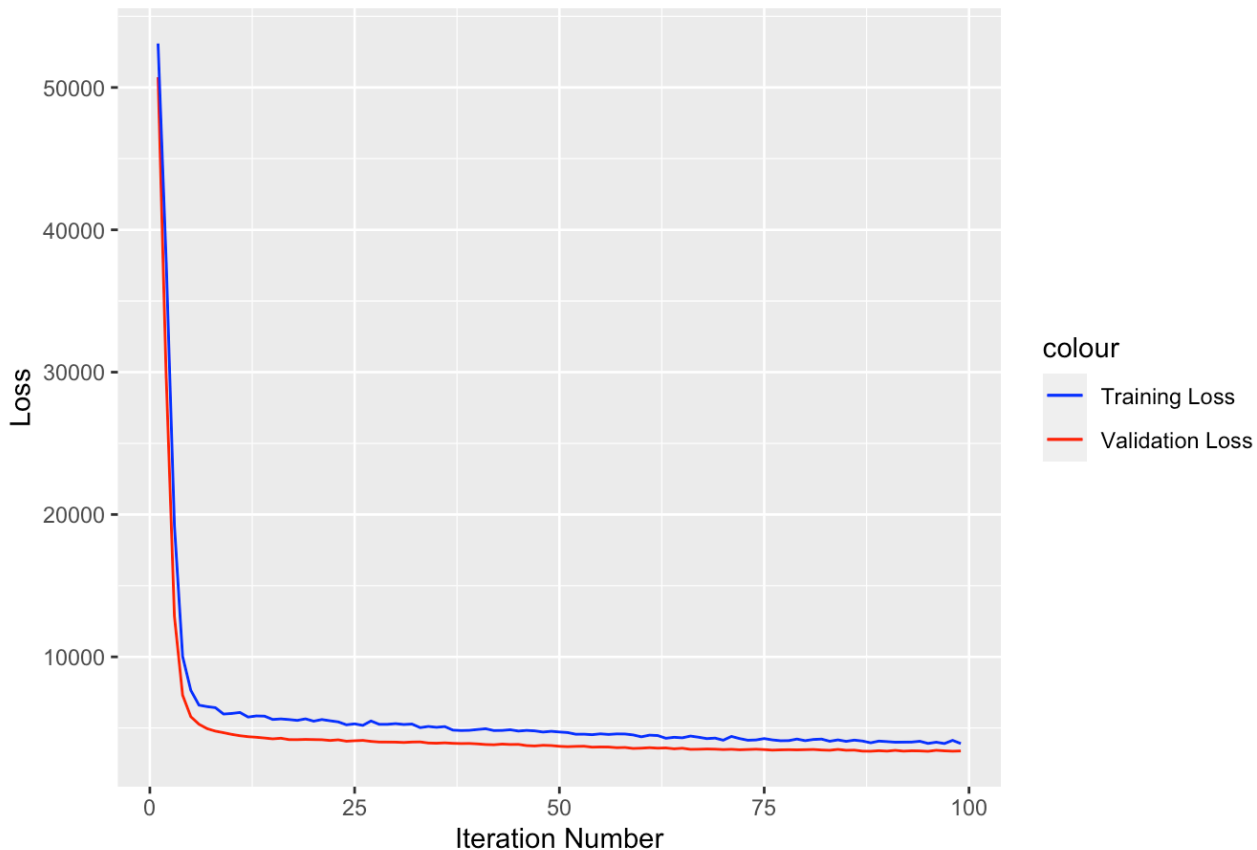
Schätzung des neuronalen Netzes

```
# Definition der Kosten-(Loss-)Funktion und der Optimierungsfunktion mit seinen Hyperparametern
model.compile(loss="mse", optimizer=Adam(lr=0.001))
# Schätzung des Modells
history = model.fit(r.training_features, r.training_labels, epochs=100,
                    validation_data = (r.validation_features, r.validation_labels), verbose=0
)
# Ggf. Speichern des geschätzten Modells
model.save("python_model.h5")
```

Auswertung der Modelloptimierung

```
# Grafische Ausgabe der Modelloptimierung
# create data
data <- data.frame(val_loss = unlist(py$history$history$val_loss),
                  loss = unlist(py$history$history$loss))
# Plot
ggplot(data[-1,]) +
  geom_line( aes(x=1:length(val_loss), y=val_loss, colour = "Validation Loss" )) +
  geom_line( aes(x=1:length(loss), y=loss, colour = "Training Loss" )) +
  scale_colour_manual( values = c("Training Loss"="blue", "Validation Loss"="red") ) +
  labs(title="Loss Function Values During Optimization") +
  xlab("Iteration Number") +
  ylab("Loss")
```

Loss Function Values During Optimization



(Ggf.) Laden eines gespeicherten Neuronalen Netzes

```
model = tf.keras.models.load_model("python_model.h5")
```

Auswertung der Schätzergebnisse

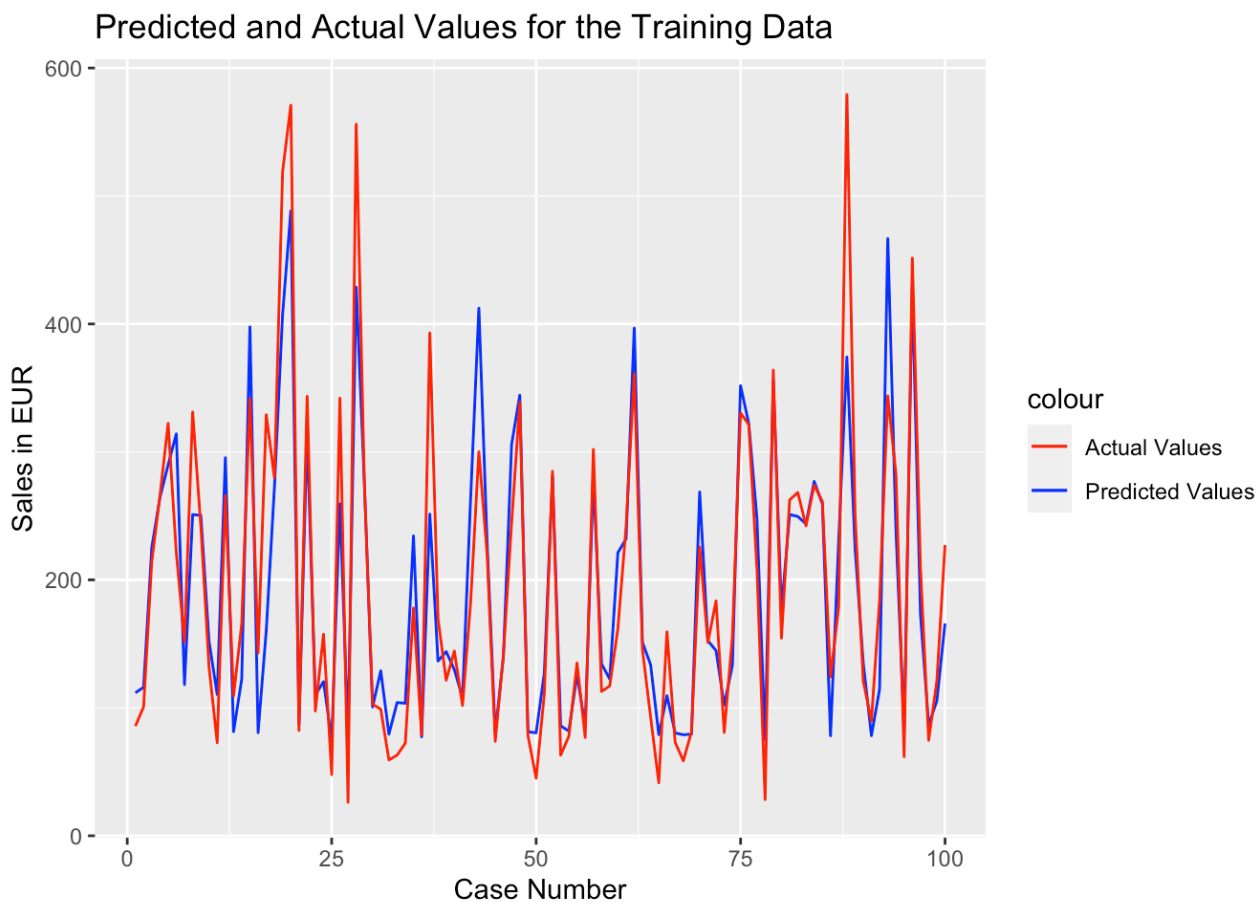
```
# Schätzung der (normierten) Preise für die Trainings- und Testdaten
training_predictions <- py$model$predict(training_features)
validation_predictions <- py$model$predict(validation_features)
validation_true_predictions <- py$model$predict(pred_data_features)
# Vergleich der Gütekriterien für die Trainings- und Testdaten
cat(paste0("MAPE on the Training Data:\t", format(mape(training_labels[[1]], as.numeric(training_predictions))*100, digits=3, nsmall=2)))
```

```
## MAPE on the Training Data: 22.75
```

```
cat(paste0("\nMAPE on the Validation Data:\t", format(mape(validation_labels[[1]], validation_predictions)*100, digits=3, nsmall=2)))
```

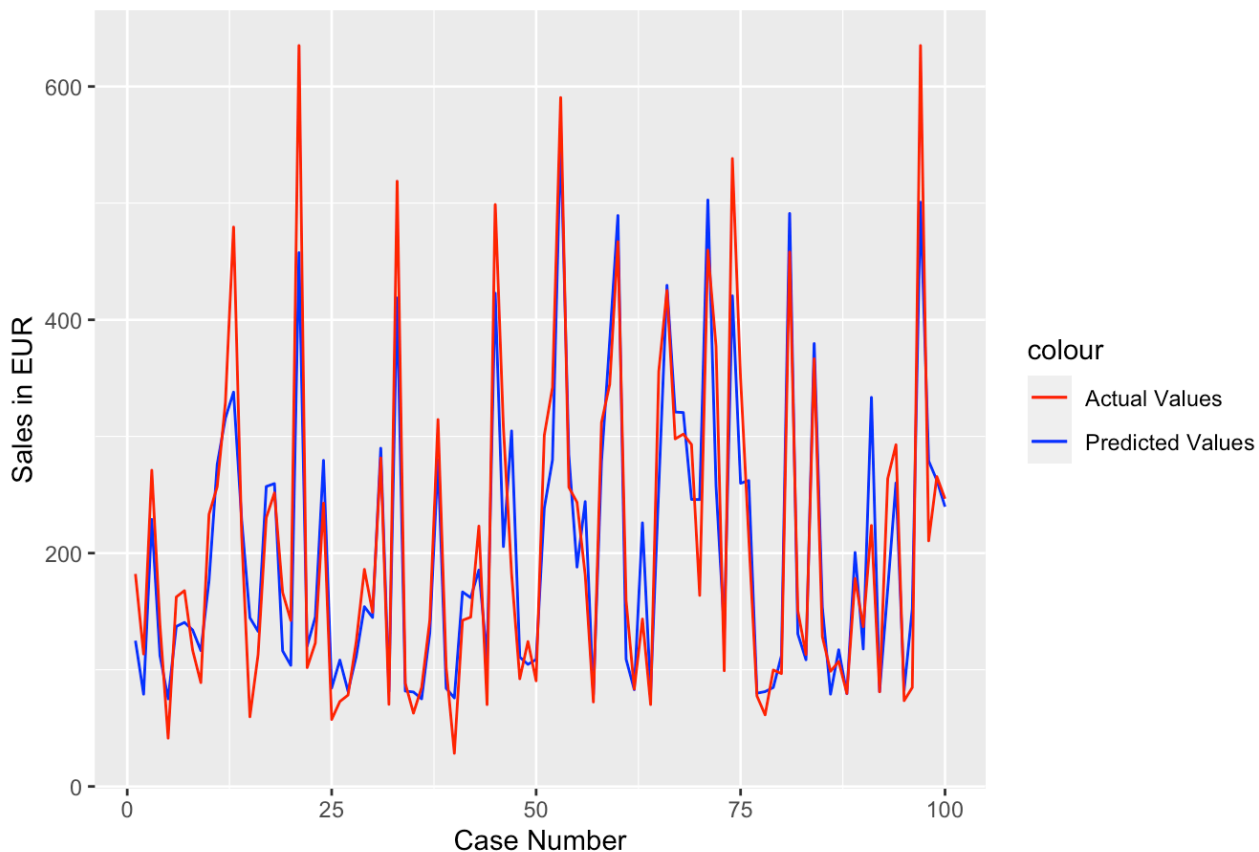
```
##
## MAPE on the Validation Data: 22.82
```

```
## Grafischer vergleich der vorhergesagten und der tatsächlichen Preise für die Trainings- und Testdaten
# Zusammenstellung der Daten für die Plots
data_train <- data.frame(prediction = training_predictions, actual = training_labels[[1]])
data_test <- data.frame(prediction = validation_predictions, actual = validation_labels[[1]])
data_pred <- data.frame(prediction = validation_true_predictions)
# Plot der Ergebnisse der Trainingsdaten
ggplot(data_train[1:100,]) +
  geom_line( aes(x=1:length(prediction), y=prediction, colour = "Predicted Values" )) +
  geom_line( aes(x=1:length(actual), y=actual, colour = "Actual Values" )) +
  scale_colour_manual( values = c("Predicted Values"="blue", "Actual Values"="red") ) +
  labs(title="Predicted and Actual Values for the Training Data") +
  xlab("Case Number") +
  ylab("Sales in EUR")
```



```
# Plot der Ergebnisse der Validierungsdaten
ggplot(data_test[1:100,]) +
  geom_line( aes(x=1:length(prediction), y=prediction, colour = "Predicted Values" )) +
  geom_line( aes(x=1:length(actual), y=actual, colour = "Actual Values" )) +
  scale_colour_manual( values = c("Predicted Values"="blue", "Actual Values"="red") ) +
  labs(title="Predicted and Actual Values for the Test Data") +
  xlab("Case Number") +
  ylab("Sales in EUR")
```

Predicted and Actual Values for the Test Data



Prädiktion 4 (Neuronales Netz)

```
pred_nn <- dplyr::bind_cols(pred_data2, as.data.frame(data_pred)) #zusammenfügen

pred_nn<- rename(pred_nn,Prediction_Umsatz_nn = prediction)#Variablenbenennung anpassen

#Faktoren wieder schick machen
pred_nn$Wochentag <- factor(weekdays(pred_nn$Datum),
levels=ordentlicheWoche)
pred_nn$Warengruppe <- factor(pred_nn$Warengruppe,
levels = c(1,2,3,4,5,6),
labels = Warengruppen)

anzeige <-dplyr::filter(pred_nn,Datum==ymd(20190607))
anzeige <-dplyr::select(anzeige, c("Warengruppe","Prediction_Umsatz_nn"))

pred_nn
```



```

##      Warengruppe      Datum Wochentag KielerWoche Bewoelkung Temperatur
## 1      Brot 2019-06-07   Freitag          0           1      17.4500
## 2      Brot 2019-06-09   Sonntag          0           5      18.6500
## 3      Brot 2019-01-09  Mittwoch          0           7       3.1375
## 4    Broetchen 2019-06-07   Freitag          0           1      17.4500
## 5    Broetchen 2019-06-09   Sonntag          0           5      18.6500
## 6    Broetchen 2019-01-09  Mittwoch          0           7       3.1375
## 7    Croissant 2019-06-07   Freitag          0           1      17.4500
## 8    Croissant 2019-06-09   Sonntag          0           5      18.6500
## 9    Croissant 2019-01-09  Mittwoch          0           7       3.1375
## 10 Konditorei 2019-06-07   Freitag          0           1      17.4500
## 11 Konditorei 2019-06-09   Sonntag          0           5      18.6500
## 12 Konditorei 2019-01-09  Mittwoch          0           7       3.1375
## 13      Kuchen 2019-06-07   Freitag          0           1      17.4500
## 14      Kuchen 2019-06-09   Sonntag          0           5      18.6500
## 15      Kuchen 2019-01-09  Mittwoch          0           7       3.1375
## 16 Saisonbrote 2019-06-07   Freitag          0           1      17.4500
## 17 Saisonbrote 2019-06-09   Sonntag          0           5      18.6500
## 18 Saisonbrote 2019-01-09  Mittwoch          0           7       3.1375
##      Windgeschwindigkeit istFeiertag Prediction_Umsatz_nn
## 1              10           0      138.47377
## 2              10           0      115.90719
## 3              22           0      101.70894
## 4              10           0      403.77881
## 5              10           0      525.39404
## 6              22           0      294.73840
## 7              10           0      164.17165
## 8              10           0      272.51883
## 9              22           0      104.86682
## 10             10           0       83.66629
## 11             10           0      120.83968
## 12             22           0       78.30128
## 13             10           0      273.58771
## 14             10           0      338.00912
## 15             22           0      219.27246
## 16             10           0       75.03831
## 17             10           0      106.38608
## 18             22           0       75.03831

```

Die Vorhersage für den ersten Tag (2019-06-07) nach Ende der Datenreihe ist:

```

##      Warengruppe Prediction_Umsatz_nn
## 1      Brot      138.47377
## 2    Broetchen      403.77881
## 3    Croissant      164.17165
## 4 Konditorei       83.66629
## 5      Kuchen      273.58771
## 6 Saisonbrote       75.03831

```

Grafischer Vergleich der Vorhersagen

```

##      Warengruppe      Datum Wochentag KielerWoche Bewoelkung Temperatur
## 1      Brot 2019-06-07   Freitag          0           1      17.4500
## 2      Brot 2019-06-09   Sonntag          0           5      18.6500

```

## 3	Brot	2019-01-09	Mittwoch	0	7	3.1375
## 4	Broetchen	2019-06-07	Freitag	0	1	17.4500
## 5	Broetchen	2019-06-09	Sonntag	0	5	18.6500
## 6	Broetchen	2019-01-09	Mittwoch	0	7	3.1375
## 7	Croissant	2019-06-07	Freitag	0	1	17.4500
## 8	Croissant	2019-06-09	Sonntag	0	5	18.6500
## 9	Croissant	2019-01-09	Mittwoch	0	7	3.1375
## 10	Konditorei	2019-06-07	Freitag	0	1	17.4500
## 11	Konditorei	2019-06-09	Sonntag	0	5	18.6500
## 12	Konditorei	2019-01-09	Mittwoch	0	7	3.1375
## 13	Kuchen	2019-06-07	Freitag	0	1	17.4500
## 14	Kuchen	2019-06-09	Sonntag	0	5	18.6500
## 15	Kuchen	2019-01-09	Mittwoch	0	7	3.1375
## 16	Saisonbrote	2019-06-07	Freitag	0	1	17.4500
## 17	Saisonbrote	2019-06-09	Sonntag	0	5	18.6500
## 18	Saisonbrote	2019-01-09	Mittwoch	0	7	3.1375
##	Windgeschwindigkeit	istFeiertag	Prediction_Umsatz_LM	Prediction_Umsatz_SVM		
## 1	10	0	135.20687	139.29890		
## 2	10	0	188.08222	102.46227		
## 3	22	0	73.24846	119.77695		
## 4	10	0	404.86622	373.23100		
## 5	10	0	457.74157	444.58441		
## 6	22	0	342.90781	323.16086		
## 7	10	0	175.32563	146.96562		
## 8	10	0	228.20098	246.05492		
## 9	22	0	113.36722	124.89878		
## 10	10	0	98.38697	76.52544		
## 11	10	0	151.26232	124.46991		
## 12	22	0	36.42856	75.97449		
## 13	10	0	285.93224	261.45276		
## 14	10	0	338.80759	276.18899		
## 15	22	0	223.97383	253.99799		
## 16	10	0	100.12620	93.58963		
## 17	10	0	153.00155	157.11996		
## 18	22	0	38.16779	76.49333		
##	Prediction_Umsatz_SVMtune	Prediction_Umsatz_nn				
## 1	143.17491	138.47377				
## 2	100.03611	115.90719				
## 3	121.13788	101.70894				
## 4	376.98689	403.77881				
## 5	488.99056	525.39404				
## 6	344.50752	294.73840				
## 7	143.61208	164.17165				
## 8	217.82495	272.51883				
## 9	139.26023	104.86682				
## 10	84.67616	83.66629				
## 11	121.64906	120.83968				
## 12	83.53556	78.30128				
## 13	270.14418	273.58771				
## 14	303.90408	338.00912				
## 15	260.70306	219.27246				
## 16	89.08484	75.03831				
## 17	107.44911	106.38608				
## 18	70.82903	75.03831				

