# Die lineare Regression mit ARMA-Fehlern - Aufgabe 1

```
library(ggplot2)
library(grid)
library(rlist)
library(forecast)
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
library(anytime)
library(ggplot2)
library(Metrics)
```

```
##
## Attaching package: 'Metrics'

## The following object is masked from 'package:forecast':
##
##     accuracy
```

```
library(regclass)
```

```
## Loading required package: bestglm

## Loading required package: leaps

## Loading required package: VGAM

## Loading required package: stats4

## Loading required package: splines

## Loading required package: rpart

## Loading required package: randomForest

## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin

## Important regclass change from 1.3:
## All functions that had a . in the name now have an _
## all.correlations -> all_correlations, cor.demo -> cor_demo, etc.
```

```
library(tseries)
library(tidytext)
```

Programmcode für das Modell der linearen Regression mit ARMA-Fehlern in Aufgabe 1 - Vorhersage der Datenrate in Up- und Downlink

## Hilfsfunktionen

```
plot_acf <- function(throughputs, type = c("acf", "pacf"),
                     title="Autokorrelationsfunktionen"){

  grid.newpage()
  pushViewport(viewport(
    layout=grid.layout(3,2, heights = unit(c(1, 5, 5), "null"))))

  if (type == "acf") {
    chosen_func <- ggAcf
    grid.text(title, gp=gpar(fontsize=20),
              vp = viewport(layout.pos.row = 1, layout.pos.col = 1:2))
    }
  else {
    chosen_func <- ggPacf
    grid.text(title, gp=gpar(fontsize=20),
              vp = viewport(layout.pos.row = 1, layout.pos.col = 1:2))
    }

  vodafone_plot <- chosen_func(throughputs$vodafone) +
    ggtitle("Vodafone") + ylab("Korrelation") + theme_grey(base_size = 16)
  tmobile_plot <- chosen_func(throughputs$tmobile) + ggtitle("T-Mobile") +
    ylab("Korrelation") + theme_grey(base_size = 16)
  o2_plot <- chosen_func(throughputs$o2) + ggtitle("O2") + ylab("Korrelation") +
    theme_grey(base_size = 16)

  print(vodafone_plot, vp=viewport(layout.pos.row = 2, layout.pos.col = 1))
  print(tmobile_plot, vp=viewport(layout.pos.row = 2, layout.pos.col = 2))
  print(o2_plot, vp=viewport(layout.pos.row = 3, layout.pos.col = 1))

}
```

## Uplink

Daten einlesen und nach Providern aufteilen

```r
ul_data = read.csv("../datasets/dataset_ul.csv", header = TRUE, sep=",", dec=".")
ul_data <- na.omit(ul_data)
ul_data$scenario <- factor(ul_data$scenario)

vodafone <- ul_data[ul_data$provider == "vodafone", ]
tmobile <- ul_data[ul_data$provider == "tmobile", ]
o2 <- ul_data[ul_data$provider == "o2", ]
providers <- list("vodafone" = vodafone, "tmobile" = tmobile, "o2" = o2)
```

Separiere Features

```r
features <- c("throughput_mbits", "payload_mb", "f_mhz", "rsrp_dbm", "rsrq_db", "rssnr_db",
              "cqi", "ta", "velocity_mps", "drive_id", "enodeb")
lm_features <- c("throughput_mbits", "payload_mb", "f_mhz", "rsrp_dbm", "rsrq_db", "rssnr_db",
                 "cqi", "ta", "velocity_mps", "enodeb")
```

Aufteilung der Daten in Test und Training

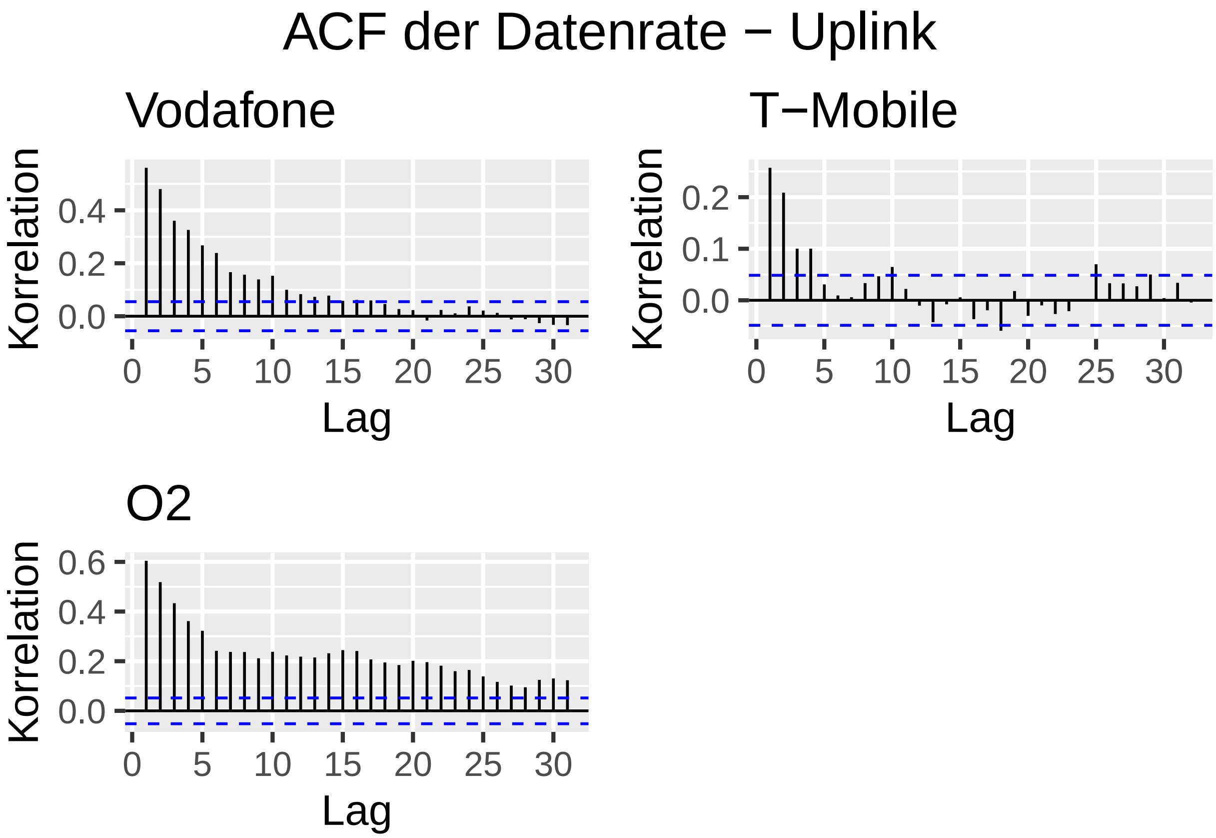
```r
train <- lapply(providers, function(provider)
  provider[
    provider["drive_id"] != 8 & provider["drive_id"] != 9 &
      provider["drive_id"] != 10, features])
test <- lapply(providers, function(provider)
  provider[
    provider["drive_id"] == 8 | provider["drive_id"] == 9 |
      provider["drive_id"] == 10, features])
```

Separiere numerische Features

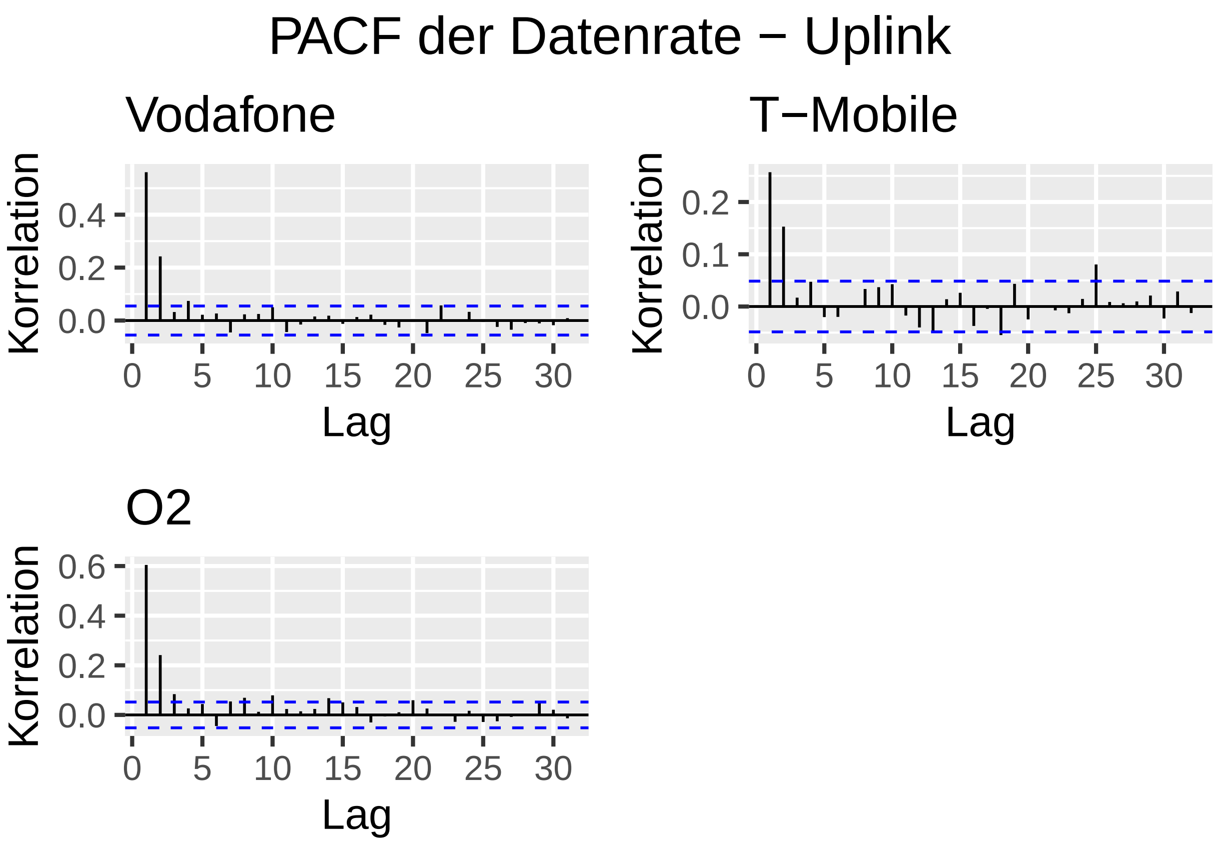
```r
numeric_features <- lm_features[as.vector(unlist(lapply(train[[1]][, lm_features],
                                                        is.numeric)))]
```

ACF und PACF von “throughput_mbits”

```r
throughputs <- list(vodafone = train$vodafone$throughput_mbits,
                    tmobile = train$tmobile$throughput_mbits,
                    o2 = train$o2$throughput_mbits)
plot_acf(throughputs, type = "acf", title = "ACF der Datenrate - Uplink")
```




```r
plot_acf(throughputs, type = "pacf", title = "PACF der Datenrate - Uplink")
```




Test auf Stationarität: Augmented Dickey-Fuller Test

```
for (j in c("vodafone", "o2", "tmobile")){
  print(j)
  for (i in numeric_features){
    adf.test(train[[j]][,i])$p.value
    print(i)
    print(adf.test(train[[j]][,i])$p.value)
  }
}
```

```
## [1] "vodafone"
## [1] "throughput_mbits"
## [1] 0.01
## [1] "payload_mb"
## [1] 0.01
## [1] "f_mhz"
## [1] 0.01
## [1] "rsrp_dbm"
## [1] 0.01
## [1] "rsrq_db"
## [1] 0.01
## [1] "rssnr_db"
## [1] 0.01
## [1] "cqi"
## [1] 0.01
```

```
## [1] "ta"
## [1] 0.01
## [1] "velocity_mps"
## [1] 0.01
## [1] "enodeb"
## [1] 0.01
## [1] "o2"
## [1] "throughput_mbits"
## [1] 0.01
## [1] "payload_mb"
## [1] 0.01
## [1] "f_mhz"
## [1] 0.01
## [1] "rsrp_dbm"
## [1] 0.01
## [1] "rsrq_db"
## [1] 0.01
## [1] "rssnr_db"
## [1] 0.01
## [1] "cqi"
## [1] 0.01
## [1] "ta"
## [1] 0.01
## [1] "velocity_mps"
## [1] 0.01
## [1] "enodeb"
## [1] 0.01
## [1] "tmobile"
## [1] "throughput_mbits"
## [1] 0.01
## [1] "payload_mb"
## [1] 0.01
## [1] "f_mhz"
## [1] 0.04507937
## [1] "rsrp_dbm"
## [1] 0.01
## [1] "rsrq_db"
## [1] 0.01
## [1] "rssnr_db"
## [1] 0.01
## [1] "cqi"
## [1] 0.01
## [1] "ta"
## [1] 0.01
## [1] "velocity_mps"
## [1] 0.01
## [1] "enodeb"
## [1] 0.01
```

Skalieren der Daten

```
for (provider in c("vodafone", "tmobile", "o2")){
  scaled <- scale(train[[provider]][, numeric_features])
  train[[provider]][, numeric_features] <- scaled
```

```r
  attr(train[[provider]], "scaled:center") <- attr(scaled, "scaled:center")
  attr(train[[provider]], "scaled:scale") <- attr(scaled, "scaled:scale")
  test[[provider]][, numeric_features] <- scale(test[[provider]][, numeric_features],
                                                center = attr(scaled, "scaled:center"),
                                                scale = attr(scaled, "scaled:scale"))
}
```

Test auf Multikollinearität mit dem VIF

```r
lm_vodafone <- lm(throughput_mbits ~ ., data = train[["vodafone"]][, lm_features])
VIF(lm_vodafone)
```

```
##   payload_mb        f_mhz     rsrp_dbm      rsrq_db     rssnr_db          cqi
##     1.007286     1.448486     2.645958     2.394323     2.781840     2.052896
##           ta velocity_mps       enodeb
##     1.380622     1.129506     1.204673
```

```r
lm_tmobile <- lm(throughput_mbits ~ ., data = train[["tmobile"]][, lm_features])
VIF(lm_tmobile)
```

```
##   payload_mb        f_mhz     rsrp_dbm      rsrq_db     rssnr_db          cqi
##     1.003522     1.255074     2.020052     2.214526     2.621853     1.841608
##           ta velocity_mps       enodeb
##     1.271389     1.276391     1.291011
```

```r
lm_o2 <- lm(throughput_mbits ~ ., data = train[["o2"]][, lm_features])
VIF(lm_o2)
```

```
##   payload_mb        f_mhz     rsrp_dbm      rsrq_db     rssnr_db          cqi
##     1.007618     1.503881     1.805449     2.809361     3.436228     2.710389
##           ta velocity_mps       enodeb
##     1.229184     1.210381     1.047898
```

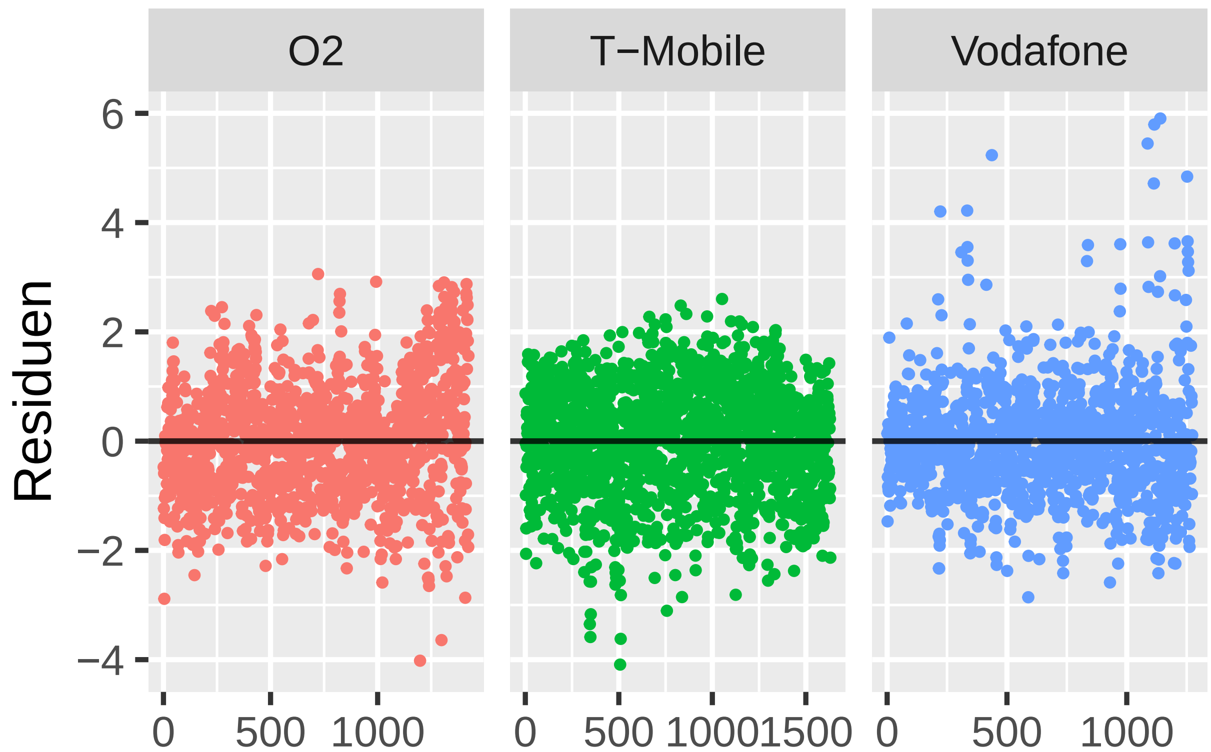
Überprüfen der Normalverteilungsannahme der Residuen

```r
res_tmobile <- data.frame(res = rstandard(lm_tmobile),
                          provider = "T-Mobile",
                          id = 1:length(rstandard(lm_tmobile)))
res_vodafone <- data.frame(res = rstandard(lm_vodafone),
                           provider = "Vodafone",
                           id = 1:length(rstandard(lm_vodafone)))
res_o2 <- data.frame(res = rstandard(lm_o2),
                     provider = "O2",
                     id = 1:length(rstandard(lm_o2)))

res_data <- rbind(res_vodafone, res_tmobile, res_o2)
```

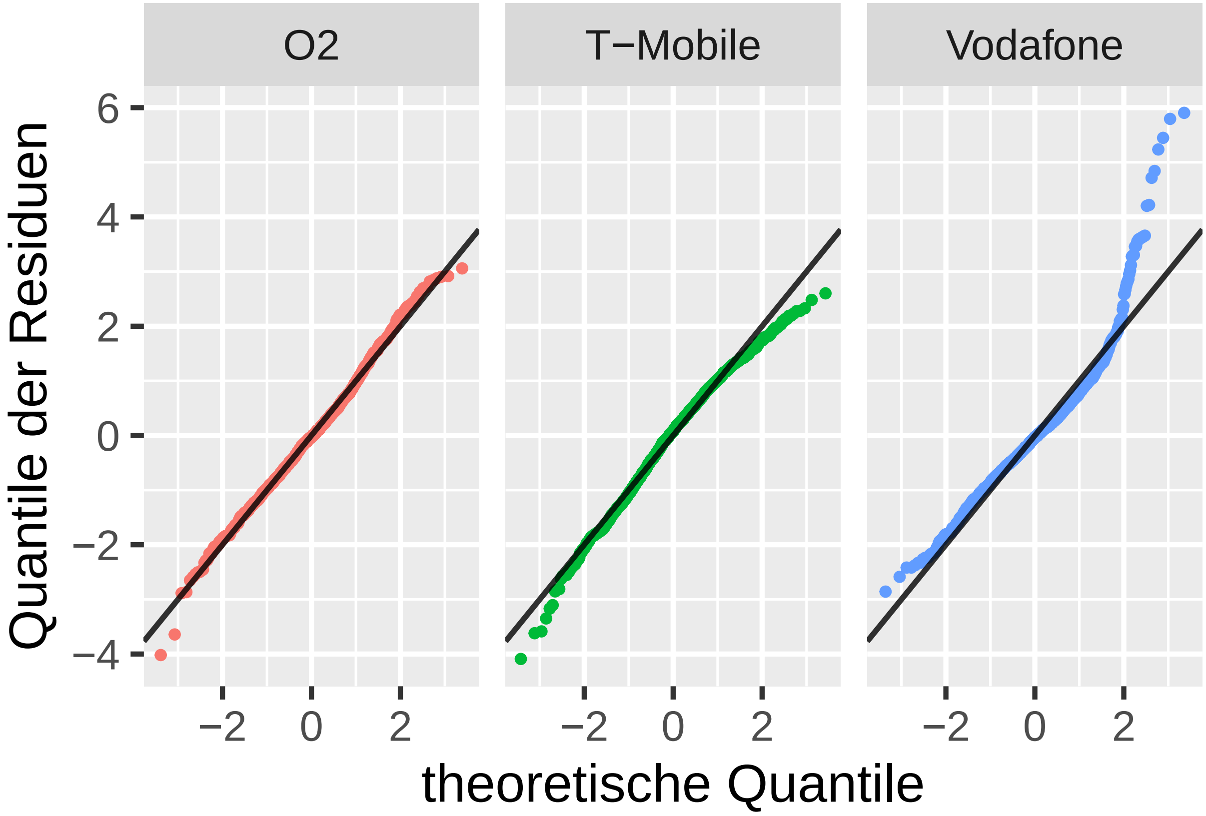
mit QQ-Plots

```r
ggplot(res_data, aes(x = id, y = res, color = provider)) + geom_point() +
  geom_abline(slope = 0, color = "black", size = 1, alpha = 0.8) +
  facet_wrap(~provider, scales = "free_x") +
  xlab("") + ylab("Residuen") +
  theme_grey(base_size = 20) +
  theme(legend.position = "none")
```




mit Scatterplots

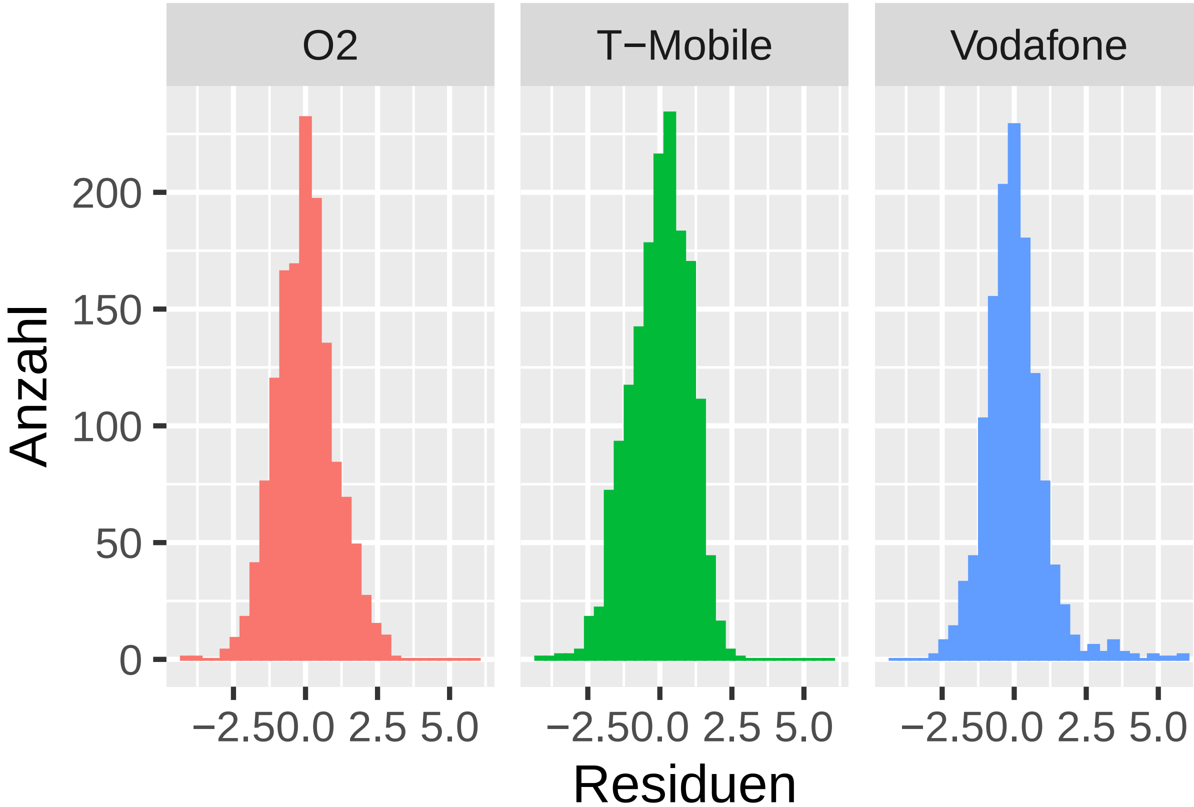
```r
ggplot(res_data, aes(sample=res, color = provider)) +
  geom_qq() +
  geom_abline(intercept = 0, slope = 1, color = "black", size = 1, alpha = 0.8) +
  facet_wrap(~provider) +
  xlab("theoretische Quantile") +
  ylab("Quantile der Residuen" ) +
  theme_grey(base_size = 20) +
  theme(legend.position = "none")
```




mit Histogrammen

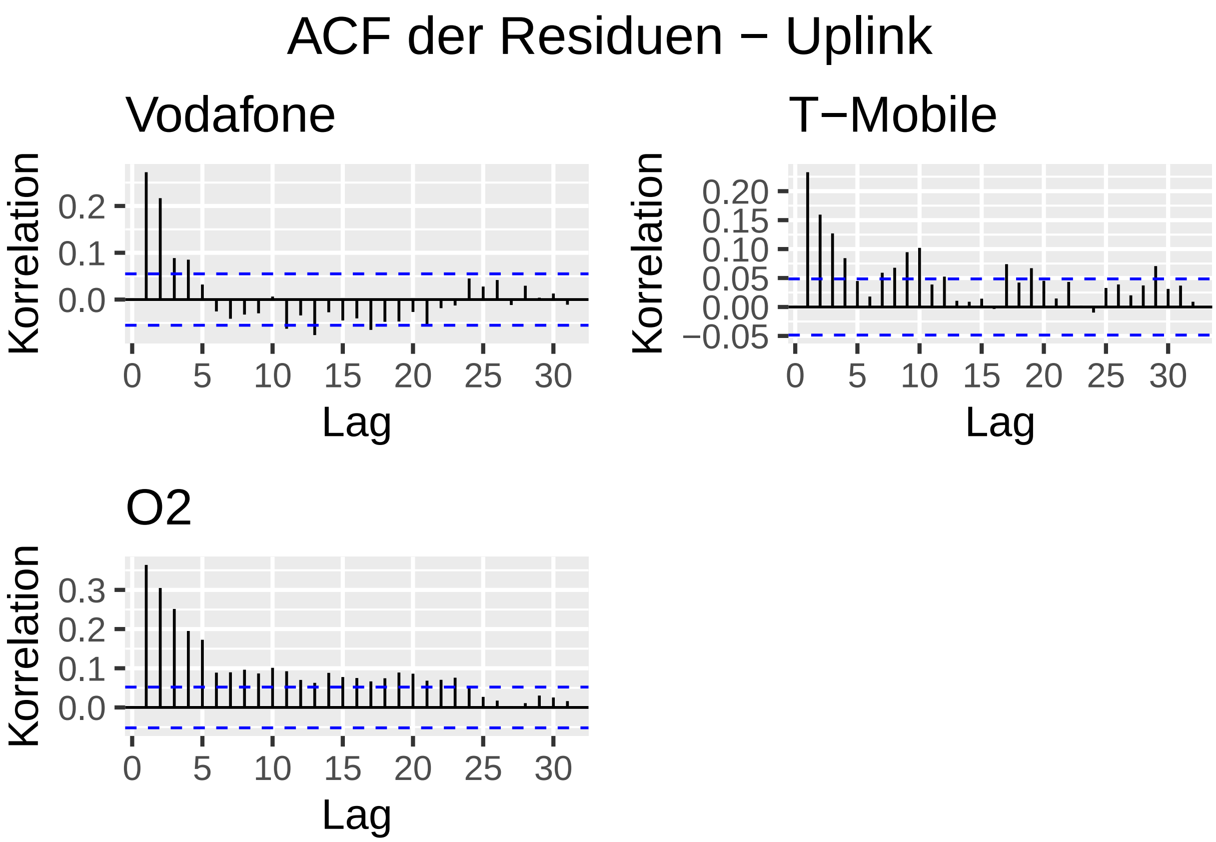
```
ggplot(res_data, aes(x = res, color = provider, fill = provider)) +
  geom_histogram() +
  facet_wrap(~ provider) +
  xlab("Residuen") + ylab("Anzahl") +
  theme_grey(base_size = 20) +
  theme(legend.position = "none")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

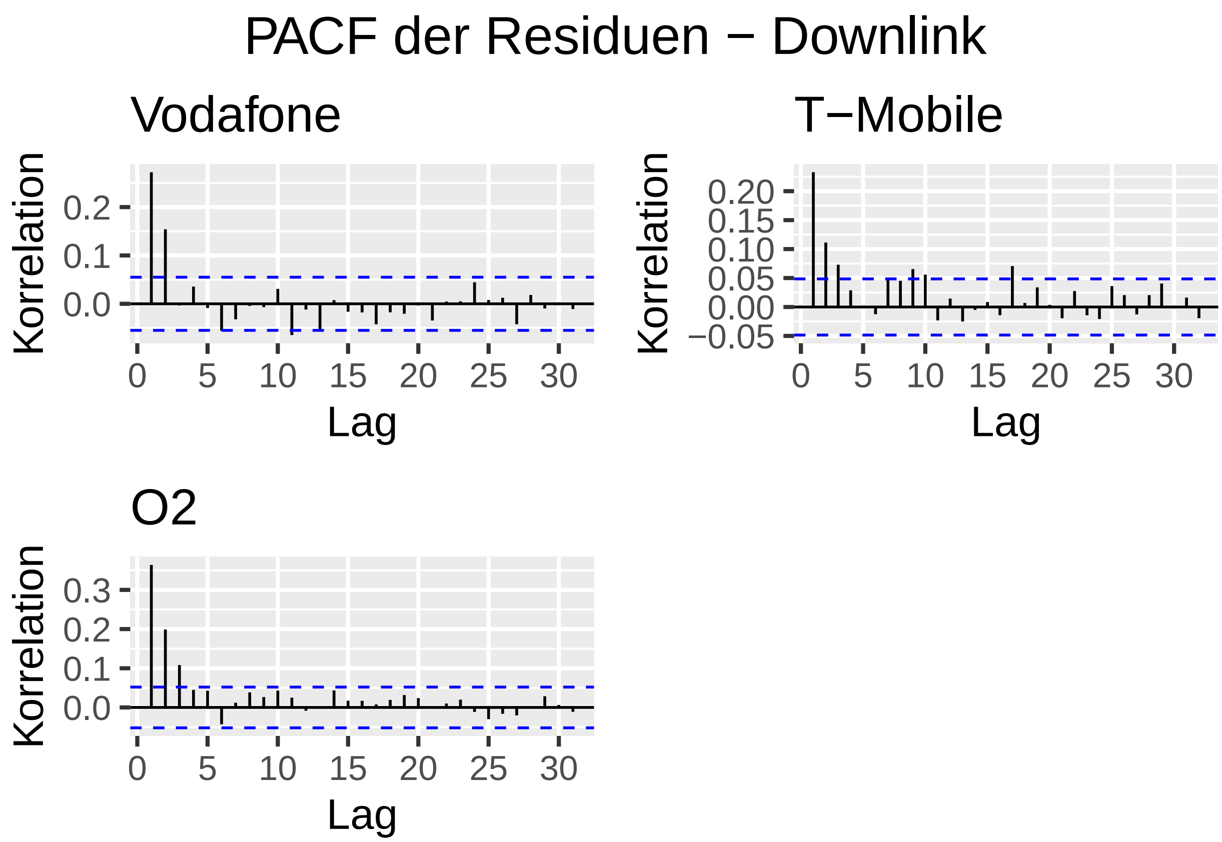



ACF und PACF der Residuen um das Grid zu bestimmen

```
plot_data <- list(vodafone = lm_vodafone$residuals,
                  tmobile = lm_tmobile$residuals,
                  o2 = lm_o2$residuals)
plot_acf(plot_data, type = "acf", title = "ACF der Residuen - Uplink")
```




```r
plot_acf(plot_data, type = "pacf", title = "PACF der Residuen - Downlink")
```




Grid (p,q): VODAFONE: (0-2,0-4), O2: (0-6,0-5) und T-MOBILE: (0-6,0-6)

```
max_ar <- 2
max_ma <- 4
nrow = (max_ar+1)*(max_ma+1)
grid_vodafone <- matrix(data = c(rep(0:max_ar, each=max_ma+1), rep(0, nrow), rep(0:max_ma, max_ar+1)),
                        nrow = nrow, ncol = 3)

max_ar <- 6
max_ma <- 5
nrow = (max_ar+1)*(max_ma+1)
grid_o2 <- matrix(data = c(rep(0:max_ar, each=max_ma+1), rep(0, nrow), rep(0:max_ma, max_ar+1)),
                  nrow = nrow, ncol = 3)

max_ar <- 6
max_ma <- 6
nrow = (max_ar+1)*(max_ma+1)
grid_tmobile <- matrix(data = c(rep(0:max_ar, each=max_ma+1), rep(0, nrow), rep(0:max_ma, max_ar+1)),
                       nrow = nrow, ncol = 3)

grids <- list("vodafone" = grid_vodafone,
              "tmobile" = grid_tmobile,
              "o2" = grid_o2)
```

Kennzahlen: MSE, MAE, Rsquared, AIC

```
vodafone_kennzahlen <- list("mse" = data.frame(),
                            "mae" = data.frame(),
                            "rsquared" = data.frame(),
                            "aic" = data.frame())

tmobile_kennzahlen <- list("mse" = data.frame(),
                           "mae" = data.frame(),
                           "rsquared" = data.frame(),
                           "aic" = data.frame())
o2_kennzahlen <- list("mse" = data.frame(),
                      "mae" = data.frame(),
                      "rsquared" = data.frame(),
                      "aic" = data.frame())
kennzahlen <- list("vodafone" = vodafone_kennzahlen,
                   "tmobile" = tmobile_kennzahlen,
                   "o2" = o2_kennzahlen,
                   "aic" = data.frame())
```

Erzeugen der Kennzahlen für die verschiedenen Provider und Testfahrten mit Zeitreihenkreuzvalidierung, sodass Fahrten 3:7 jeweils Test - 1 -> 1:(test_id-1) Training

```
for (provider in c("vodafone", "tmobile", "o2")){

  cv_train <- train[[provider]][
    train[[provider]]["drive_id"] == 1 | train[[provider]]["drive_id"] == 2,
    lm_features
  ]

  all_mse <- data.frame(
    matrix(rep(NA, 5*nrow(grids[[provider]])), nrow = nrow(grids[[provider]])),
    row.names = as.character(1:nrow(grids[[provider]]))

  )
  colnames(all_mse) <- c(paste("test_id", as.character(3:7), sep="_"))

  all_mae <- data.frame(
    matrix(rep(NA, 5*nrow(grids[[provider]])), nrow=nrow(grids[[provider]])),
    row.names = as.character(1:nrow(grids[[provider]]))
  )
  colnames(all_mae) <- c(paste("test_id", as.character(3:7), sep="_"))

  all_rsquared <- data.frame(
    matrix(rep(NA, 5*nrow(grids[[provider]])), nrow=nrow(grids[[provider]])),
    row.names = as.character(1:nrow(grids[[provider]]))
  )
  colnames(all_rsquared) <- c(paste("test_id", as.character(3:7), sep="_"))

  all_aic <- data.frame(
    matrix(rep(NA, 5*nrow(grids[[provider]])), nrow=nrow(grids[[provider]])),
    row.names = as.character(1:nrow(grids[[provider]]))
  )
  colnames(all_aic) <- c(paste("test_id", as.character(3:7), sep="_"))
```

```
  for (test_id in 3:7){

    if(test_id > 3){
      cv_train <- rbind(cv_train,
                        train[[provider]][
                          train[[provider]]["drive_id"] == test_id-1, lm_features
                        ])
    }
    cv_test <- train[[provider]][train[[provider]]["drive_id"] == test_id, lm_features]



    for (row in 1:nrow(grids[[provider]])){

      y <- ts(cv_train[, "throughput_mbits"])
      xreg <- cv_train[, lm_features[-which(lm_features == "throughput_mbits")]]

      xreg <- data.matrix(xreg)
      arima_fit <- Arima(y = y, order = grids[[provider]][row,], xreg = xreg, method = "ML")

      y <- ts(cv_test[, "throughput_mbits"])
      xreg <- cv_test[, lm_features[-which(lm_features == "throughput_mbits")]]

      xreg <- data.matrix(xreg)
      pred <- forecast(arima_fit, xreg = xreg)

      all_mse[row, paste("test_id", test_id, sep = "_")] <- mse(unclass(y), unclass(pred$mean))
      all_mae[row, paste("test_id", test_id, sep = "_")] <- mae(unclass(y), unclass(pred$mean))
      all_rsquared[row, paste("test_id", test_id, sep = "_")] <- 1 -
        sum((unclass(pred$mean)-unclass(y))^2)/sum((mean(unclass(y))-unclass(y))^2)
      all_aic[row, paste("test_id", test_id, sep = "_")] <- pred$model$aic
    }
    kennzahlen[[provider]]$mse <- all_mse
    kennzahlen[[provider]]$mae <- all_mae
    kennzahlen[[provider]]$rsquared <- all_rsquared
    kennzahlen[[provider]]$aic <- all_aic

  }
}
```

Suche für jeden Provider die Kombination heraus, welche die besten Kennzahlen erzeugt

```
## [1] 0 0 2
```

```
## [1] 0 0 1
```

```
## [1] 0 0 2
```

```
## [1] 2 0 2
```

```
## [1] 0 0 0
```

```
## [1] 0 0 0
```

```
## [1] 0 0 0
```

```
## [1] 5 0 6
```

```
## [1] 0 0 0
```

```
## [1] 0 0 0
```

```
## [1] 0 0 0
```

```
## [1] 3 0 1
```

```
## $vodafone
## [1] 0 0 2
##
## $tmobile
## [1] 0 0 0
##
## $o2
## [1] 0 0 0
```

Modell für den kompletten Trainingsdatensatz fitten und für Test predicten und Predictions zurücktransformieren

```
kennzahlen_final <- list("vodafone" = list(),
                         "tmobile" = list(),
                         "o2" = list())
predictions <- list("vodafone" = list(),
                    "tmobile" = list(),
                    "o2" = list())
coeff <- list("vodafone" = list(),
              "tmobile" = list(),
              "o2" = list())

for (provider in c("vodafone", "tmobile", "o2")){
  y <- ts(train[[provider]][, "throughput_mbits"])
  xreg <- train[[provider]][, lm_features[-which(lm_features == "throughput_mbits")]]

  xreg <- data.matrix(xreg)
  arima_fit <- Arima(y = y, order = parameter[[provider]], xreg = xreg, method = "ML")
  coeff[[provider]] <- arima_fit$coef[c("intercept",lm_features[-which(lm_features ==
                                                                         "throughput_mbits")])]
  y <- ts(test[[provider]][, "throughput_mbits"])
  xreg <- test[[provider]][, lm_features[-which(lm_features == "throughput_mbits")]]

  xreg <- data.matrix(xreg)
  predictions[[provider]] <- forecast(arima_fit, xreg = xreg)
  predictions[[provider]]$rescaled_forecast <- predictions[[provider]]$mean * attr(train[[provider]],
                                                                "scaled:scale")["throughput_mbits"]+
    attr(train[[provider]], "scaled:center")["throughput_mbits"]
```

```
  predictions[[provider]]$rescaled_y <- y * attr(train[[provider]],
                                                 "scaled:scale")["throughput_mbits"] +
    attr(train[[provider]], "scaled:center")["throughput_mbits"]
  rescaled_y <- unclass(predictions[[provider]]$rescaled_y)
  rescaled_forecast <- unclass(predictions[[provider]]$rescaled_forecast)
  kennzahlen_final[[provider]]$mse <- mse(rescaled_y, rescaled_forecast)
  kennzahlen_final[[provider]]$mae <- mae(rescaled_y, rescaled_forecast)
  kennzahlen_final[[provider]]$rsquared <- 1 - sum((rescaled_forecast-rescaled_y)^2)/
    sum((mean(rescaled_y)-rescaled_y)^2)
}
```

# Downlink

Daten einlesen und nach Providern aufteilen

```
dl_data = read.csv("../datasets/dataset_dl.csv", header = TRUE, sep=",", dec=".")
dl_data <- na.omit(dl_data)
dl_data$scenario <- factor(dl_data$scenario)

vodafone <- dl_data[dl_data$provider == "vodafone", ]
tmobile <- dl_data[dl_data$provider == "tmobile", ]
o2 <- dl_data[dl_data$provider == "o2", ]
providers <- list("vodafone" = vodafone, "tmobile" = tmobile, "o2" = o2)
```

Separiere die Features

```
features <- c("throughput_mbits", "payload_mb", "f_mhz", "rsrp_dbm", "rsrq_db", "rssnr_db",
              "cqi", "ta", "velocity_mps", "drive_id", "enodeb")
lm_features <- c("throughput_mbits", "payload_mb", "f_mhz", "rsrp_dbm", "rsrq_db", "rssnr_db",
                 "cqi", "ta", "velocity_mps", "enodeb")
```

Aufteilung der Daten in Training und Test

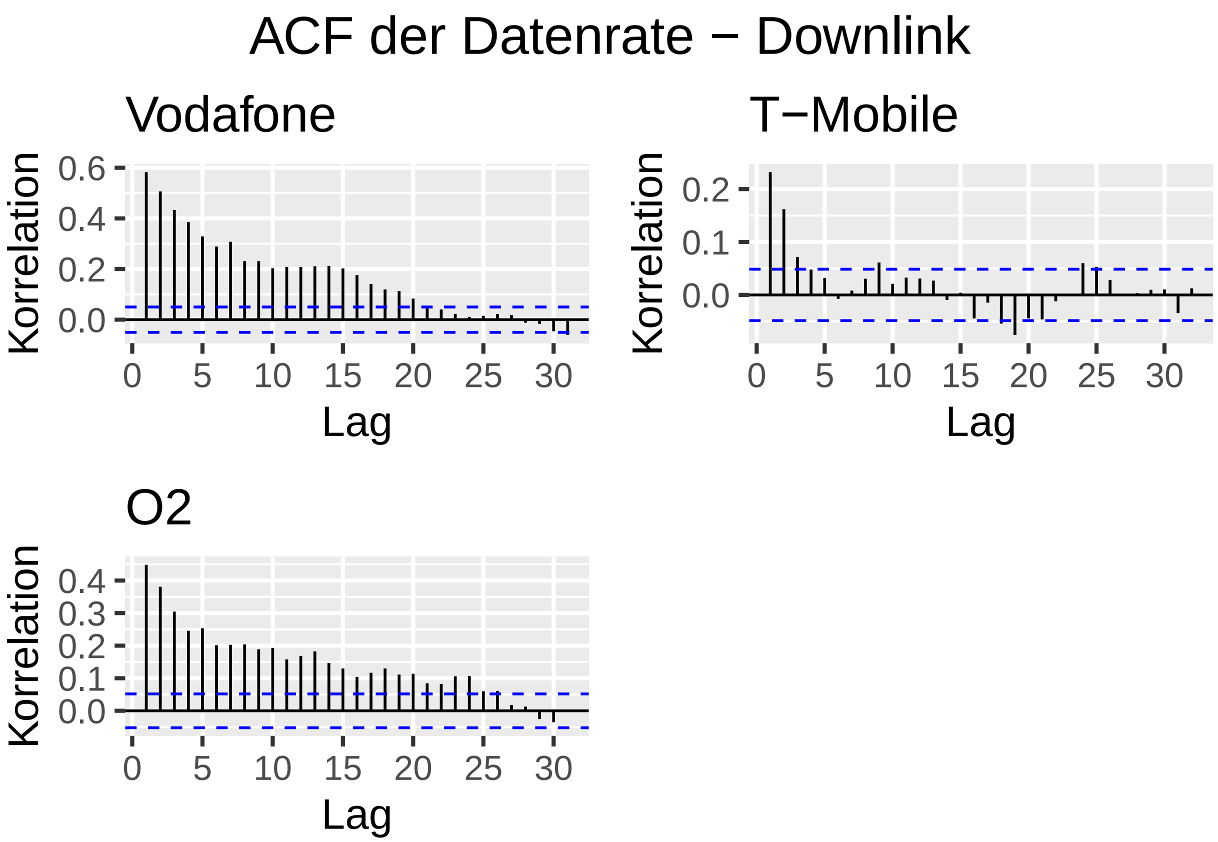
```
train <- lapply(providers, function(provider)
  provider[
    provider["drive_id"] != 8 & provider["drive_id"] != 9 &
      provider["drive_id"] != 10, features])
test <- lapply(providers, function(provider)
  provider[
    provider["drive_id"] == 8 | provider["drive_id"] == 9 |
      provider["drive_id"] == 10, features])
```

alle numerischen Features

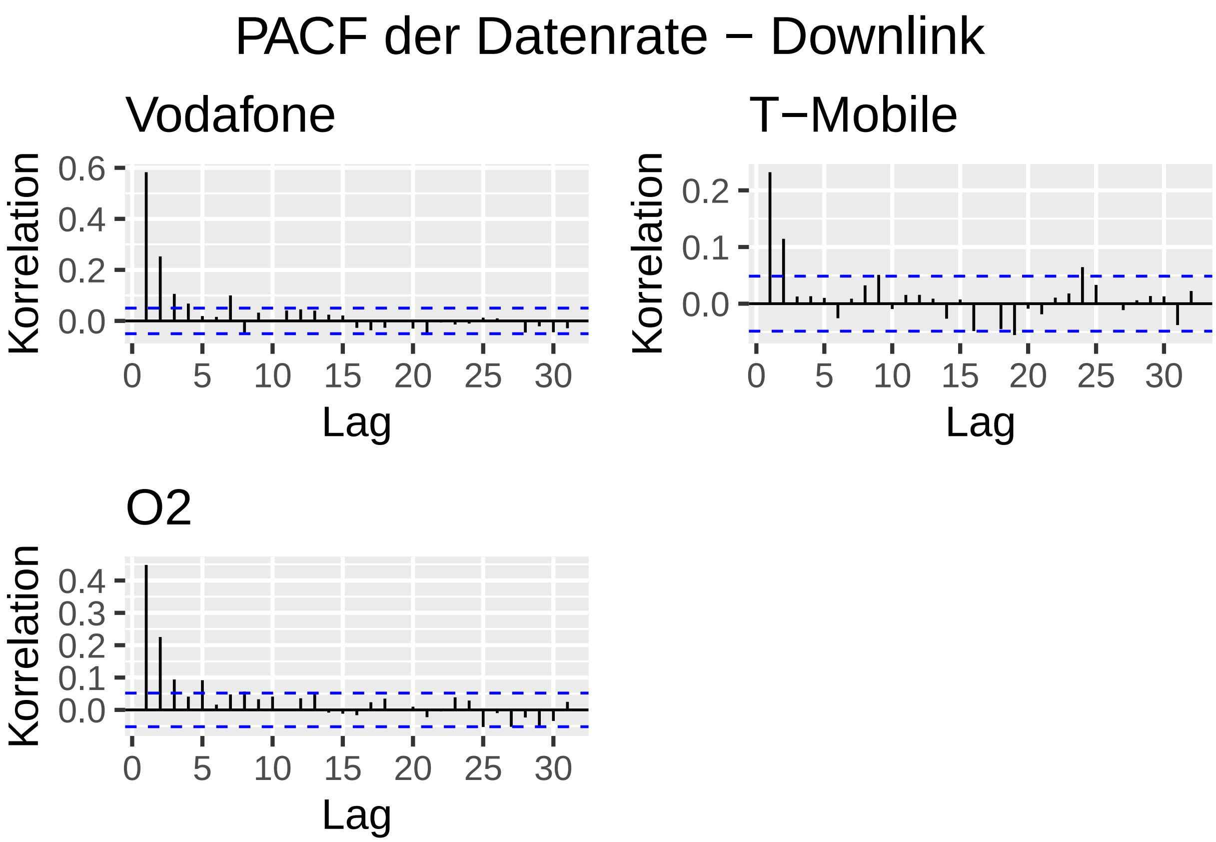
```
numeric_features <- lm_features[as.vector(unlist(lapply(train[[1]][, lm_features],
                                                     is.numeric)))]
```

ACF und PACF von "throughput_mbits"

```r
throughputs <- list(vodafone = train$vodafone$throughput_mbits,
                    tmobile = train$tmobile$throughput_mbits,
                    o2 = train$o2$throughput_mbits)
plot_acf(throughputs, type = "acf", title = "ACF der Datenrate - Downlink")
```




```r
plot_acf(throughputs, type = "pacf", title = "PACF der Datenrate - Downlink")
```




Test auf Stationarität: Augmented Dickey-Fuller Test

```
for (j in c("vodafone", "o2", "tmobile")){
  print(j)
  for (i in numeric_features){
    adf.test(train[[j]][,i])$p.value
    print(i)
    print(adf.test(train[[j]][,i])$p.value)
  }
}
```

```
## [1] "vodafone"
## [1] "throughput_mbits"
## [1] 0.01
## [1] "payload_mb"
## [1] 0.01
## [1] "f_mhz"
## [1] 0.01
## [1] "rsrp_dbm"
## [1] 0.01
## [1] "rsrq_db"
## [1] 0.01
## [1] "rssnr_db"
## [1] 0.01
## [1] "cqi"
## [1] 0.01
```

```
## [1] "ta"
## [1] 0.01
## [1] "velocity_mps"
## [1] 0.01
## [1] "enodeb"
## [1] 0.01
## [1] "o2"
## [1] "throughput_mbits"
## [1] 0.01
## [1] "payload_mb"
## [1] 0.01
## [1] "f_mhz"
## [1] 0.01
## [1] "rsrp_dbm"
## [1] 0.01
## [1] "rsrq_db"
## [1] 0.01
## [1] "rssnr_db"
## [1] 0.01
## [1] "cqi"
## [1] 0.01
## [1] "ta"
## [1] 0.01
## [1] "velocity_mps"
## [1] 0.01
## [1] "enodeb"
## [1] 0.01
## [1] "tmobile"
## [1] "throughput_mbits"
## [1] 0.01
## [1] "payload_mb"
## [1] 0.01
## [1] "f_mhz"
## [1] 0.03622455
## [1] "rsrp_dbm"
## [1] 0.01
## [1] "rsrq_db"
## [1] 0.01
## [1] "rssnr_db"
## [1] 0.01
## [1] "cqi"
## [1] 0.01
## [1] "ta"
## [1] 0.01
## [1] "velocity_mps"
## [1] 0.01
## [1] "enodeb"
## [1] 0.01
```

Skalieren der Daten

```r
for (provider in c("vodafone", "tmobile", "o2")){
  scaled <- scale(train[[provider]][, numeric_features])
  train[[provider]][, numeric_features] <- scaled
```

```r
  attr(train[[provider]], "scaled:center") <- attr(scaled, "scaled:center")
  attr(train[[provider]], "scaled:scale") <- attr(scaled, "scaled:scale")
  test[[provider]][, numeric_features] <- scale(test[[provider]][, numeric_features],
                                                center = attr(scaled, "scaled:center"),
                                                scale = attr(scaled, "scaled:scale"))
}
```

Überprüfen auf Multikollinearität mit dem VIF

```r
lm_vodafone <- lm(throughput_mbits ~ ., data = train[["vodafone"]][, lm_features])
VIF(lm_vodafone)
```

```
##   payload_mb        f_mhz     rsrp_dbm      rsrq_db     rssnr_db          cqi
##     1.004517     1.450469     2.581545     2.432827     2.635343     2.106758
##           ta velocity_mps       enodeb
##     1.298794     1.146502     1.032784
```

```r
lm_tmobile <- lm(throughput_mbits ~ ., data = train[["tmobile"]][, lm_features])
VIF(lm_tmobile)
```

```
##   payload_mb        f_mhz     rsrp_dbm      rsrq_db     rssnr_db          cqi
##     1.008777     1.245794     2.033054     2.195838     2.627988     1.859510
##           ta velocity_mps       enodeb
##     1.268326     1.276257     1.284749
```

```r
lm_o2 <- lm(throughput_mbits ~ ., data = train[["o2"]][, lm_features])
VIF(lm_o2)
```

```
##   payload_mb        f_mhz     rsrp_dbm      rsrq_db     rssnr_db          cqi
##     1.008994     1.483388     1.833720     2.810867     3.559795     2.843181
##           ta velocity_mps       enodeb
##     1.219191     1.187955     1.051398
```

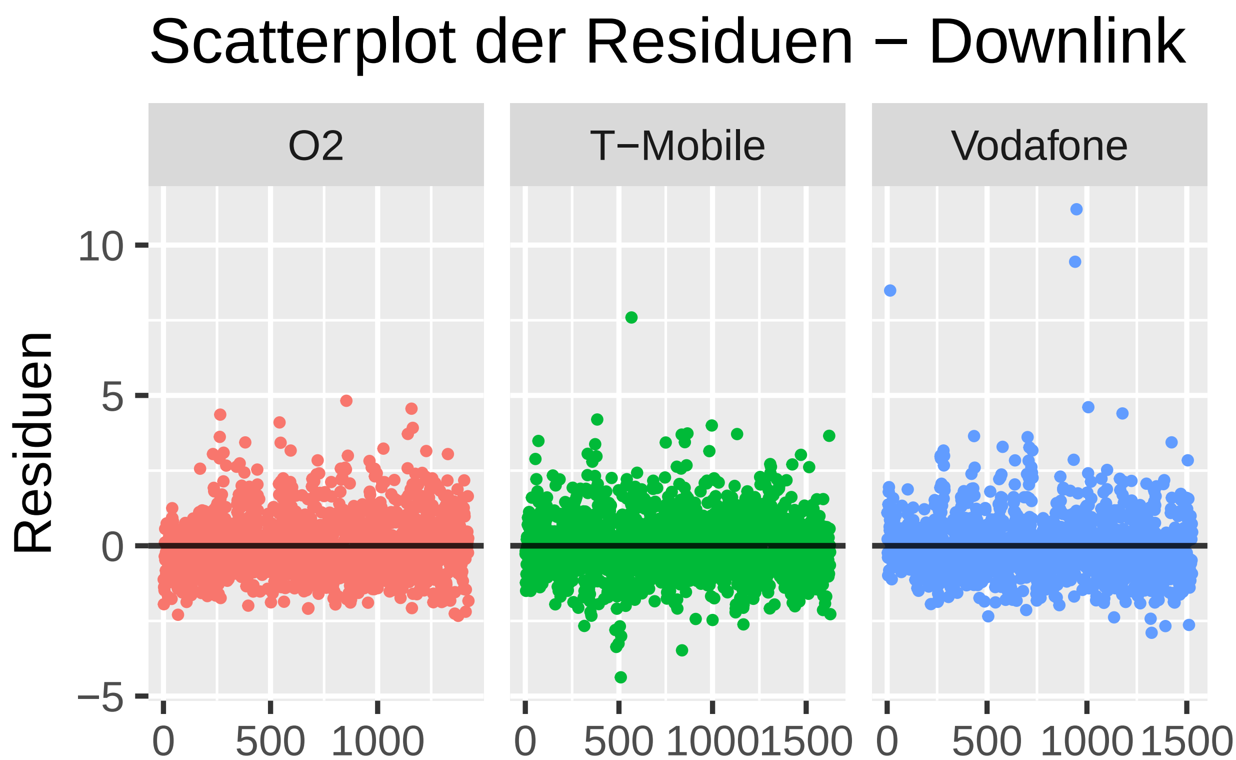
Überprüfen der Normalverteilungsannahme der Residuen

```r
res_tmobile <- data.frame(res = rstandard(lm_tmobile),
                          provider = "T-Mobile",
                          id = 1:length(rstandard(lm_tmobile)))
res_vodafone <- data.frame(res = rstandard(lm_vodafone),
                           provider = "Vodafone",
                           id = 1:length(rstandard(lm_vodafone)))
res_o2 <- data.frame(res = rstandard(lm_o2),
                     provider = "O2",
                     id = 1:length(rstandard(lm_o2)))

res_data <- rbind(res_vodafone, res_tmobile, res_o2)
```

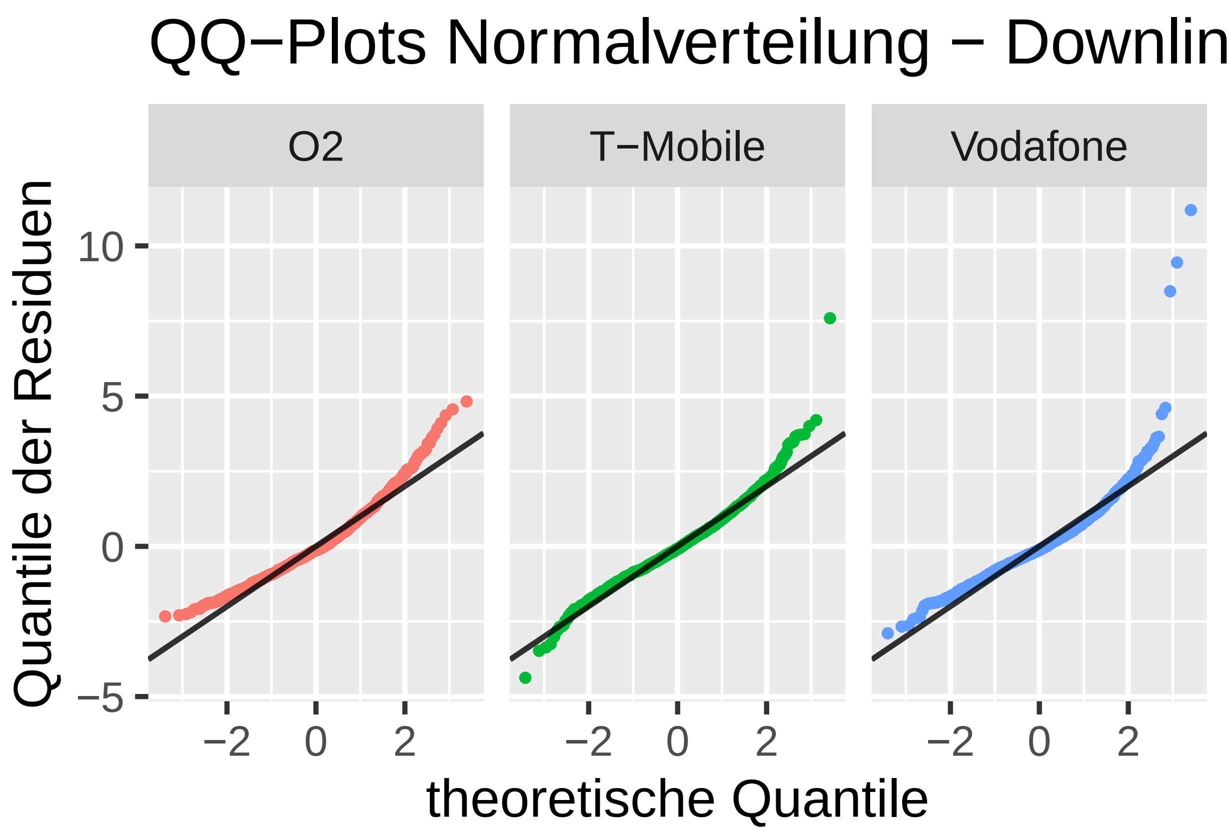
mit Scatterplots

```
ggplot(res_data, aes(x = id, y = res, color = provider)) + geom_point() +
  geom_abline(slope = 0, color = "black", size = 1, alpha = 0.8) +
  facet_wrap(~provider, scales = "free_x") +
  ggtitle("Scatterplot der Residuen - Downlink") +
  xlab("") + ylab("Residuen") +
  theme_grey(base_size = 20) +
  theme(legend.position = "none")
```




mit QQ-Plots

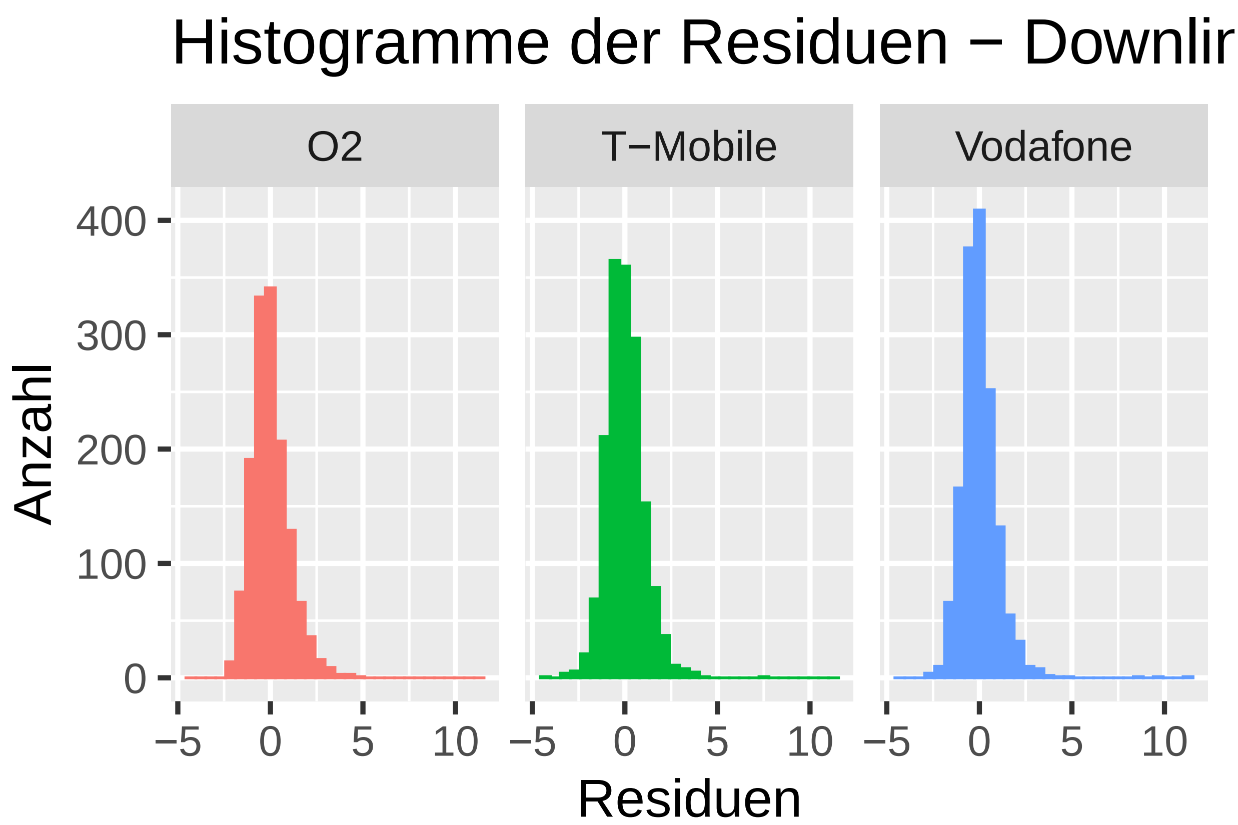
```
ggplot(res_data, aes(sample=res, color = provider)) +
  geom_qq() +
  geom_abline(intercept = 0, slope = 1, color = "black", size = 1, alpha = 0.8) +
  facet_wrap(~provider) +
  ggtitle("QQ-Plots Normalverteilung - Downlink") +
  xlab("theoretische Quantile") +
  ylab("Quantile der Residuen" ) +
  theme_grey(base_size = 20) +
  theme(legend.position = "none")
```




mit Histogrammen

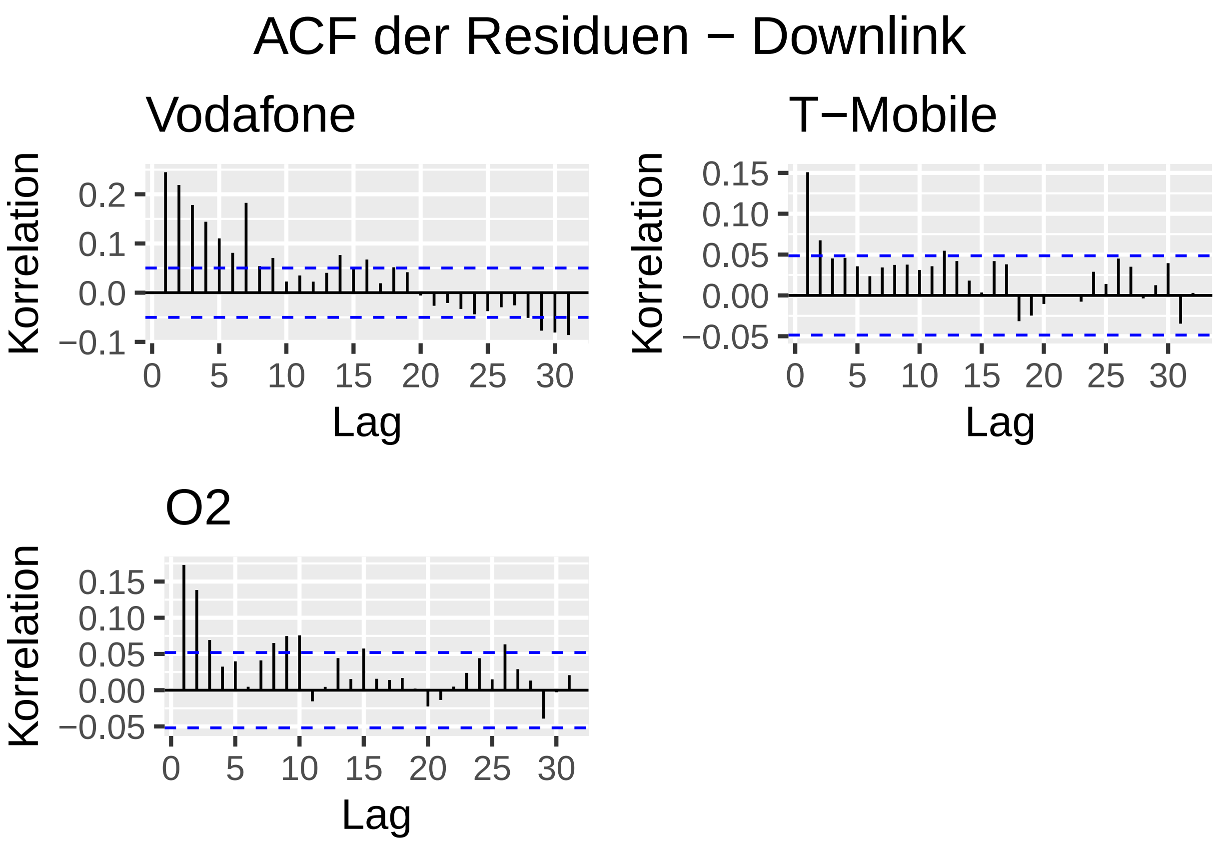
```
ggplot(res_data, aes(x = res, color = provider, fill = provider)) +
  geom_histogram() +
  facet_wrap(~ provider) +
  ggtitle("Histogramme der Residuen - Downlink") +
  xlab("Residuen") + ylab("Anzahl") +
  theme_grey(base_size = 20) +
  theme(legend.position = "none")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

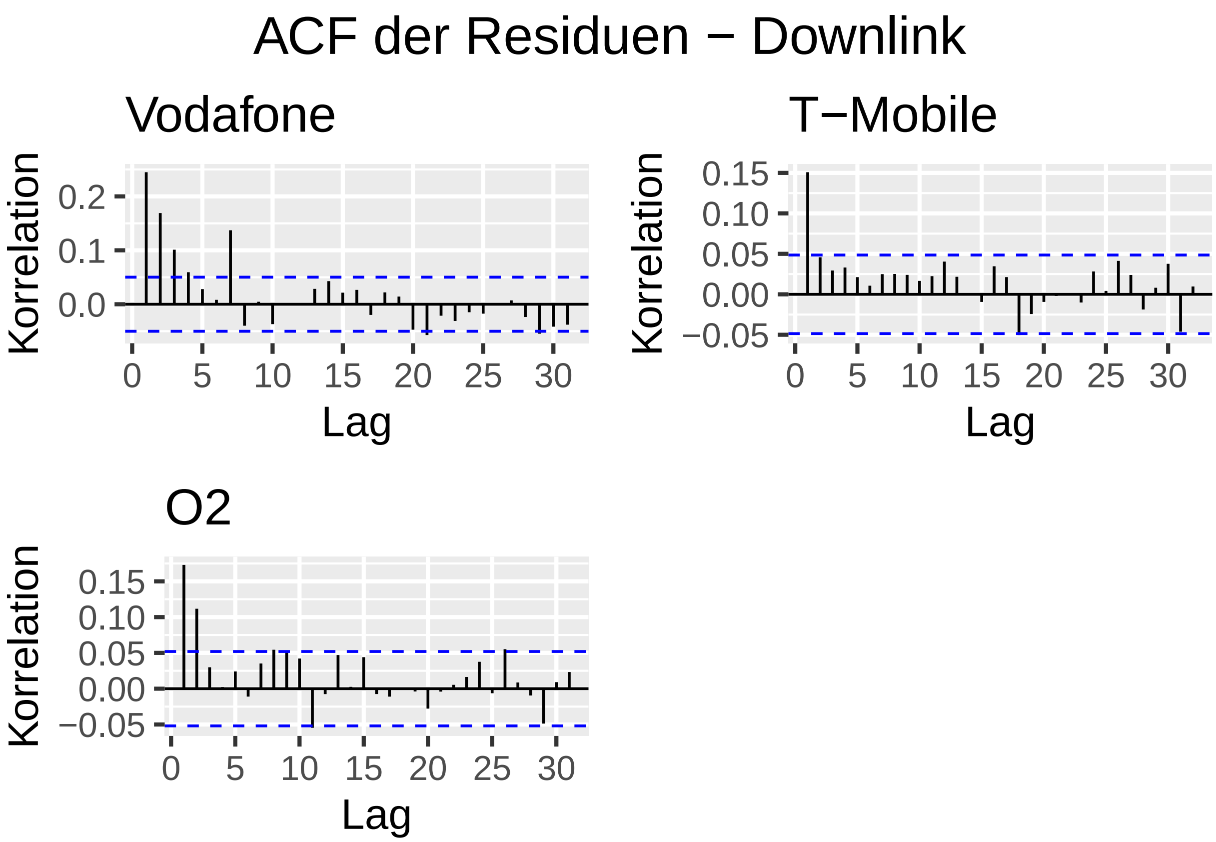



ACF und PACF der Residuen um das Grid zu bestimmen

```
plot_data <- list(vodafone = lm_vodafone$residuals,
                  tmobile = lm_tmobile$residuals,
                  o2 = lm_o2$residuals)
plot_acf(plot_data, type = "acf", title = "ACF der Residuen - Downlink")
```




```r
plot_acf(plot_data, type = "pacf", title = "ACF der Residuen - Downlink")
```




Grid (p,q): VODAFONE: (0-7,0-7), O2: (0-2,0-10) und T-MOBILE: (0-1,0-2)

```
max_ar <- 7
max_ma <- 7
nrow = (max_ar+1)*(max_ma+1)
grid_vodafone <- matrix(data = c(rep(0:max_ar, each=max_ma+1), rep(0, nrow), rep(0:max_ma, max_ar+1)),
                        nrow = nrow, ncol = 3)

max_ar <- 2
max_ma <- 10
nrow = (max_ar+1)*(max_ma+1)
grid_o2 <- matrix(data = c(rep(0:max_ar, each=max_ma+1), rep(0, nrow), rep(0:max_ma, max_ar+1)),
                        nrow = nrow, ncol = 3)

max_ar <- 1
max_ma <- 2
nrow = (max_ar+1)*(max_ma+1)
grid_tmobile <- matrix(data = c(rep(0:max_ar, each=max_ma+1), rep(0, nrow), rep(0:max_ma, max_ar+1)),
                        nrow = nrow, ncol = 3)

grids <- list("vodafone" = grid_vodafone,
              "tmobile" = grid_tmobile,
              "o2" = grid_o2)
```

Kennzahlen: MSE, MAE, Rsquared, AIC

```r
vodafone_kennzahlen <- list("mse" = data.frame(),
                            "mae" = data.frame(),
                            "rsquared" = data.frame(),
                            "aic" = data.frame())

tmobile_kennzahlen <- list("mse" = data.frame(),
                           "mae" = data.frame(),
                           "rsquared" = data.frame(),
                           "aic" = data.frame())
o2_kennzahlen <- list("mse" = data.frame(),
                      "mae" = data.frame(),
                      "rsquared" = data.frame(),
                      "aic" = data.frame())
kennzahlen <- list("vodafone" = vodafone_kennzahlen,
                   "tmobile" = tmobile_kennzahlen,
                   "o2" = o2_kennzahlen,
                   "aic" = data.frame())
```

Erzeugen der Kennzahlen für die verschiedenen Provider und Testfahrten mit Zeitreihenkreuzvalidierung, sodass Fahrten 3:7 jeweils Test - 1 -> 1:(test_id-1) Training

```r
for (provider in c("vodafone", "tmobile", "o2")){
  cv_train <- train[[provider]][
      train[[provider]]["drive_id"] == 1 | train[[provider]]["drive_id"] == 2,
      lm_features
    ]

  all_mse <- data.frame(
    matrix(rep(NA, 5*nrow(grids[[provider]])), nrow = nrow(grids[[provider]])),
    row.names = as.character(1:nrow(grids[[provider]]))
  )
  colnames(all_mse) <- c(paste("test_id", as.character(3:7), sep="_"))

  all_mae <- data.frame(
    matrix(rep(NA, 5*nrow(grids[[provider]])), nrow=nrow(grids[[provider]])),
    row.names = as.character(1:nrow(grids[[provider]]))
  )
  colnames(all_mae) <- c(paste("test_id", as.character(3:7), sep="_"))

  all_rsquared <- data.frame(
    matrix(rep(NA, 5*nrow(grids[[provider]])), nrow=nrow(grids[[provider]])),
    row.names = as.character(1:nrow(grids[[provider]]))
  )
  colnames(all_rsquared) <- c(paste("test_id", as.character(3:7), sep="_"))

  all_aic <- data.frame(
    matrix(rep(NA, 5*nrow(grids[[provider]])), nrow=nrow(grids[[provider]])),
    row.names = as.character(1:nrow(grids[[provider]]))
  )
  colnames(all_aic) <- c(paste("test_id", as.character(3:7), sep="_"))

  for (test_id in 3:7){
```

```
    if(test_id > 3){
      cv_train <- rbind(cv_train,
                        train[[provider]][
                          train[[provider]]["drive_id"] == test_id-1, lm_features
                        ])
    }
    cv_test <- train[[provider]][train[[provider]]["drive_id"] == test_id, lm_features]



    for (row in 1:nrow(grids[[provider]])){
      y <- ts(cv_train[, "throughput_mbits"])
      xreg <- cv_train[, lm_features[-which(lm_features == "throughput_mbits")]]

      xreg <- data.matrix(xreg)

      arima_fit <- Arima(y = y, order = grids[[provider]][row,], xreg = xreg, method = "ML")

      y <- ts(cv_test[, "throughput_mbits"])
      xreg <- cv_test[, lm_features[-which(lm_features == "throughput_mbits")]]
      xreg <- data.matrix(xreg)
      pred <- forecast(arima_fit, xreg = xreg)
      all_mse[row, paste("test_id", test_id, sep = "_")] <- mse(unclass(y), unclass(pred$mean))
      all_mae[row, paste("test_id", test_id, sep = "_")] <- mae(unclass(y), unclass(pred$mean))
      all_rsquared[row, paste("test_id", test_id, sep = "_")] <- 1 -
        sum((unclass(pred$mean)-unclass(y))^2)/sum((mean(unclass(y))-unclass(y))^2)
      all_aic[row, paste("test_id", test_id, sep = "_")] <- pred$model$aic
    }
    kennzahlen[[provider]]$mse <- all_mse
    kennzahlen[[provider]]$mae <- all_mae
    kennzahlen[[provider]]$rsquared <- all_rsquared
    kennzahlen[[provider]]$aic <- all_aic

  }
}
```

Suche für jeden Provider die Kombination heraus, welche die besten Kennzahlen erzeugt

```
## [1] 1 0 6
```

```
## [1] 7 0 0
```

```
## [1] 7 0 0
```

```
## [1] 7 0 2
```

```
## [1] 0 0 0
```

```
## [1] 0 0 0
```

```
## [1] 0 0 0
```

```
## [1] 1 0 2
```

```
## [1] 0 0 0
```

```
## [1] 0 0 0
```

```
## [1] 0 0 0
```

```
## [1] 2 0 1
```

```
## $vodafone
## [1] 1 0 6
##
## $tmobile
## [1] 0 0 0
##
## $o2
## [1] 0 0 0
```

Modell für den kompletten Trainingsdatensatz fitten und für Test predicten und Predictions zurücktransformieren

```
kennzahlen_final <- list("vodafone" = list(),
                   "tmobile" = list(),
                   "o2" = list())
predictions <- list("vodafone" = list(),
                    "tmobile" = list(),
                    "o2" = list())
coeff <- list("vodafone" = list(),
                     "tmobile" = list(),
                     "o2" = list())

for (provider in c("tmobile", "o2", "vodafone")){
  y <- ts(train[[provider]][, "throughput_mbits"])
  xreg <- train[[provider]][, lm_features[-which(lm_features == "throughput_mbits")]]
  xreg <- data.matrix(xreg)
  arima_fit <- Arima(y = y, order = parameter[[provider]], xreg = xreg, method = "ML")
  coeff[[provider]] <- arima_fit$coef[c("intercept",lm_features[-which(lm_features ==
                                                                  "throughput_mbits")])]
  y <- ts(test[[provider]][, "throughput_mbits"])
  xreg <- test[[provider]][, lm_features[-which(lm_features == "throughput_mbits")]]
  xreg <- data.matrix(xreg)
  predictions[[provider]] <- forecast(arima_fit, xreg = xreg)
  predictions[[provider]]$rescaled_forecast <- predictions[[provider]]$mean * attr(train[[provider]],
                                                              "scaled:scale")["throughput_mbits"]+
    attr(train[[provider]], "scaled:center")["throughput_mbits"]
  predictions[[provider]]$rescaled_y <- y * attr(train[[provider]],
                                                 "scaled:scale")["throughput_mbits"] +
    attr(train[[provider]], "scaled:center")["throughput_mbits"]
  rescaled_y <- unclass(predictions[[provider]]$rescaled_y)
  rescaled_forecast <- unclass(predictions[[provider]]$rescaled_forecast)
  kennzahlen_final[[provider]]$mse <- mse(rescaled_y, rescaled_forecast)
```

```
  kennzahlen_final[[provider]]$mae <- mae(rescaled_y, rescaled_forecast)
  kennzahlen_final[[provider]]$rsquared <- 1 - sum((rescaled_forecast-rescaled_y)^2)/
                                                sum((mean(rescaled_y)-rescaled_y)^2)
}
```