dir.create("~/Downloads/jena\_climate", recursive = TRUE)

## Warning in dir.create("~/Downloads/jena\_climate", recursive = TRUE): 'C:  
## \Users\Ghirghir\Documents\Downloads\jena\_climate' already exists

download.file(  
 "https://s3.amazonaws.com/keras-datasets/jena\_climate\_2009\_2016.csv.zip",  
 "~/Downloads/jena\_climate/jena\_climate\_2009\_2016.csv.zip"  
)  
unzip(  
 "~/Downloads/jena\_climate/jena\_climate\_2009\_2016.csv.zip",  
 exdir = "~/Downloads/jena\_climate"  
)

S. 199

library(tibble)  
library(readr)  
library(ggplot2)  
library(keras)

## Warning: package 'keras' was built under R version 3.6.2

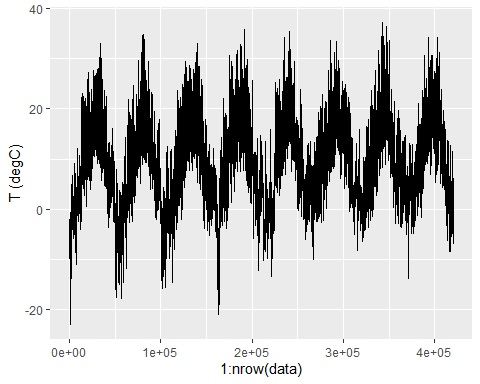
data\_dir <- "~/Downloads/jena\_climate"  
fname <- file.path(data\_dir, "jena\_climate\_2009\_2016.csv")  
data <- read\_csv(fname)

## Parsed with column specification:  
## cols(  
## `Date Time` = col\_character(),  
## `p (mbar)` = col\_double(),  
## `T (degC)` = col\_double(),  
## `Tpot (K)` = col\_double(),  
## `Tdew (degC)` = col\_double(),  
## `rh (%)` = col\_double(),  
## `VPmax (mbar)` = col\_double(),  
## `VPact (mbar)` = col\_double(),  
## `VPdef (mbar)` = col\_double(),  
## `sh (g/kg)` = col\_double(),  
## `H2OC (mmol/mol)` = col\_double(),  
## `rho (g/m\*\*3)` = col\_double(),  
## `wv (m/s)` = col\_double(),  
## `max. wv (m/s)` = col\_double(),  
## `wd (deg)` = col\_double()  
## )

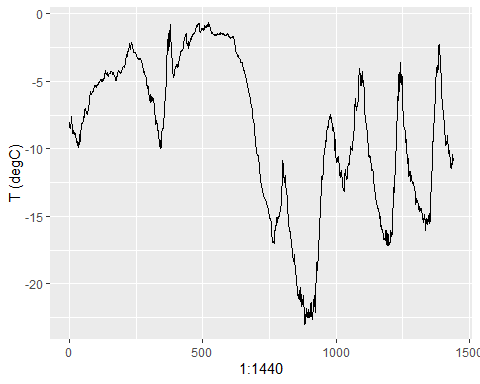
glimpse(data)

## Observations: 420,551  
## Variables: 15  
## $ `Date Time` <chr> "01.01.2009 00:10:00", "01.01.2009 00:20:00"...  
## $ `p (mbar)` <dbl> 996.52, 996.57, 996.53, 996.51, 996.51, 996....  
## $ `T (degC)` <dbl> -8.02, -8.41, -8.51, -8.31, -8.27, -8.05, -7...  
## $ `Tpot (K)` <dbl> 265.40, 265.01, 264.91, 265.12, 265.15, 265....  
## $ `Tdew (degC)` <dbl> -8.90, -9.28, -9.31, -9.07, -9.04, -8.78, -8...  
## $ `rh (%)` <dbl> 93.3, 93.4, 93.9, 94.2, 94.1, 94.4, 94.8, 94...  
## $ `VPmax (mbar)` <dbl> 3.33, 3.23, 3.21, 3.26, 3.27, 3.33, 3.44, 3....  
## $ `VPact (mbar)` <dbl> 3.11, 3.02, 3.01, 3.07, 3.08, 3.14, 3.26, 3....  
## $ `VPdef (mbar)` <dbl> 0.22, 0.21, 0.20, 0.19, 0.19, 0.19, 0.18, 0....  
## $ `sh (g/kg)` <dbl> 1.94, 1.89, 1.88, 1.92, 1.92, 1.96, 2.04, 2....  
## $ `H2OC (mmol/mol)` <dbl> 3.12, 3.03, 3.02, 3.08, 3.09, 3.15, 3.27, 3....  
## $ `rho (g/m\*\*3)` <dbl> 1307.75, 1309.80, 1310.24, 1309.19, 1309.00,...  
## $ `wv (m/s)` <dbl> 1.03, 0.72, 0.19, 0.34, 0.32, 0.21, 0.18, 0....  
## $ `max. wv (m/s)` <dbl> 1.75, 1.50, 0.63, 0.50, 0.63, 0.63, 0.63, 0....  
## $ `wd (deg)` <dbl> 152.3, 136.1, 171.6, 198.0, 214.3, 192.7, 16...

ggplot(data, aes(x = 1:nrow(data), y = `T (degC)`)) + geom\_line()



ggplot(data[1:1440,], aes(x = 1:1440, y = `T (degC)`)) + geom\_line()



Preparing the data

data <- data.matrix(data[,-1])  
  
train\_data <- data[1:200000,]  
mean <- apply(train\_data, 2, mean)  
std <- apply(train\_data, 2, sd)  
data <- scale(data, center = mean, scale = std)

data <- scale(data, center = mean, scale = std)

generator <- function(data, lookback, delay, min\_index, max\_index,  
 shuffle = FALSE, batch\_size = 128, step = 6) {  
 if (is.null(max\_index))  
 max\_index <- nrow(data) - delay - 1  
 i <- min\_index + lookback  
 function() {  
 if (shuffle) {  
 rows <- sample(c((min\_index+lookback):max\_index), size = batch\_size)  
 } else {  
 if (i + batch\_size >= max\_index)  
 i <<- min\_index + lookback  
 rows <- c(i:min(i+batch\_size-1, max\_index))  
 i <<- i + length(rows)  
 }  
   
 samples <- array(0, dim = c(length(rows),   
 lookback / step,  
 dim(data)[[-1]]))  
 targets <- array(0, dim = c(length(rows)))  
   
 for (j in 1:length(rows)) {  
 indices <- seq(rows[[j]] - lookback, rows[[j]] - 1,   
 length.out = dim(samples)[[2]])  
 samples[j,,] <- data[indices,]  
 targets[[j]] <- data[rows[[j]] + delay,2]  
 }   
   
 list(samples, targets)  
 }  
}

lookback <- 1440  
step <- 6  
delay <- 144  
batch\_size <- 128  
train\_gen <- generator(  
data,  
lookback = lookback,  
delay = delay,  
min\_index = 1,  
max\_index = 200000,  
shuffle = TRUE,  
step = step,  
batch\_size = batch\_size  
)  
val\_gen = generator(  
data,  
lookback = lookback,  
delay = delay,  
min\_index = 200001,  
max\_index = 300000,  
step = step,  
batch\_size = batch\_size  
)  
test\_gen <- generator(  
data,  
lookback = lookback,  
delay = delay,  
min\_index = 300001,  
max\_index = NULL,  
step = step,  
batch\_size = batch\_size  
)  
val\_steps <- (300000 - 200001 - lookback) / batch\_size  
test\_steps <- (nrow(data) - 300001 - lookback) / batch\_size

A common sense, non-machine learning baseline

## mean(abs(preds - targets))  
  
evaluate\_naive\_method <- function() {  
 batch\_maes <- c()  
 for (step in 1:val\_steps) {  
 c(samples, targets) %<-% val\_gen()  
 preds <- samples[,dim(samples)[[2]],2]  
 mae <- mean(abs(preds - targets))  
 batch\_maes <- c(batch\_maes, mae)  
 }  
 print(mean(batch\_maes))  
}

celsius\_mae <- 0.29 \* std[[2]]

A basic machine learning approach

model <- keras\_model\_sequential() %>%   
 layer\_flatten(input\_shape = c(lookback / step, dim(data)[-1])) %>%   
 layer\_dense(units = 32, activation = "relu") %>%   
 layer\_dense(units = 1)

model %>% compile(  
 optimizer = optimizer\_rmsprop(),  
 loss = "mae"  
)  
history <- model %>% fit\_generator(  
 train\_gen,  
 steps\_per\_epoch = 500,  
 epochs = 20,  
 validation\_data = val\_gen,  
 validation\_steps = val\_steps  
)

plot(history)

