

# Assignment 2

Group 4

2024-09-16

```
options(contrasts = c("contr.sum", "contr.poly"))
require("ggplot2")
require("dplyr")
require("ppcor")
require("caret")
require("tidyverse")
require("stringr")
require("lubridate")
require("tsibble")
require("ggfortify")
require("gridExtra")
require("reshape2")

library(imputeTS)    # Time series missing value imputation

library(jsonlite)    # handle JSON data returned by Frost
library(tidyr)       # unpack data from JSON format
library(tidyverse)   # data manipulation with mutate etc, string formatting
library(lubridate)   # process date and time information
library(tsibble)     # special tibbles for time series
library(fpp3)        # autoplot() and gg_season() for time series
library(readr)        # to read the Frost client ID from file
```

## Task 1: Dimension reduction on air quality data

### Part A: Get

- Obtain data from <https://archive.ics.uci.edu/dataset/360/air+quality>.
- Provide a brief description of the data based on the information from the website.

```
airquality <- read.table("AirQualityUCI.csv", sep=";", dec=",", header= T)

airqual <- airquality %>%
  dplyr::select(-c("X", "X.1")) %>%
  na.omit() %>%
  mutate(timedate = dmy_hms(paste(Date, Time))) %>%
  dplyr::select(-c("Time", "Date")) %>%
  relocate(timedate) %>%
  as_tibble()

summary(airqual)
```

##      timedate

CO.GT.

PT08.S1.CO.

```

## Min.   :2004-03-10 18:00:00   Min.   :-200.00   Min.   :-200
## 1st Qu.:2004-06-16 05:00:00   1st Qu.:  0.60   1st Qu.: 921
## Median :2004-09-21 16:00:00   Median :  1.50   Median :1053
## Mean   :2004-09-21 16:00:00   Mean   :-34.21   Mean   :1049
## 3rd Qu.:2004-12-28 03:00:00   3rd Qu.:  2.60   3rd Qu.:1221
## Max.   :2005-04-04 14:00:00   Max.   : 11.90   Max.   :2040
##      NMHC.GT.          C6H6.GT.          PT08.S2.NMHC.      NOx.GT.
## Min.   :-200.0   Min.   :-200.000   Min.   :-200.0   Min.   :-200.0
## 1st Qu.:-200.0   1st Qu.:  4.000   1st Qu.: 711.0   1st Qu.: 50.0
## Median :-200.0   Median :  7.900   Median : 895.0   Median : 141.0
## Mean   :-159.1   Mean   :  1.866   Mean   : 894.6   Mean   : 168.6
## 3rd Qu.:-200.0   3rd Qu.: 13.600   3rd Qu.:1105.0   3rd Qu.: 284.0
## Max.   :1189.0   Max.   : 63.700   Max.   :2214.0   Max.   :1479.0
##      PT08.S3.NOx.        NO2.GT.          PT08.S4.NO2.        PT08.S5.03.
## Min.   :-200     Min.   :-200.00   Min.   :-200     Min.   :-200.0
## 1st Qu.: 637    1st Qu.: 53.00   1st Qu.:1185    1st Qu.: 700.0
## Median : 794    Median : 96.00   Median :1446    Median : 942.0
## Mean   : 795    Mean   : 58.15   Mean   :1391    Mean   : 975.1
## 3rd Qu.: 960    3rd Qu.:133.00   3rd Qu.:1662    3rd Qu.:1255.0
## Max.   :2683    Max.   : 340.00   Max.   :2775    Max.   :2523.0
##      T              RH              AH
## Min.   :-200.000   Min.   :-200.00   Min.   :-200.0000
## 1st Qu.: 10.900   1st Qu.: 34.10   1st Qu.: 0.6923
## Median : 17.200   Median : 48.60   Median : 0.9768
## Mean   : 9.778   Mean   : 39.49   Mean   : -6.8376
## 3rd Qu.: 24.100   3rd Qu.: 61.90   3rd Qu.: 1.2962
## Max.   : 44.600   Max.   : 88.70   Max.   : 2.2310

head(airqual)

## # A tibble: 6 x 14
##   timedate           CO.GT.  PT08.S1.CO. NMHC.GT. C6H6.GT. PT08.S2.NMHC. NOx.GT.
##   <dttm>            <dbl>    <int>    <int>    <dbl>    <int>    <int>
## 1 2004-03-10 18:00:00    2.6     1360     150     11.9     1046     166
## 2 2004-03-10 19:00:00     2      1292     112      9.4      955     103
## 3 2004-03-10 20:00:00    2.2     1402      88       9      939     131
## 4 2004-03-10 21:00:00    2.2     1376      80      9.2      948     172
## 5 2004-03-10 22:00:00    1.6     1272      51      6.5      836     131
## 6 2004-03-10 23:00:00    1.2     1197      38      4.7      750      89
## # i 7 more variables: PT08.S3.NOx. <int>, NO2.GT. <int>, PT08.S4.NO2. <int>,
## #   PT08.S5.03. <int>, T <dbl>, RH <dbl>, AH <dbl>
```

Contains the responses of a gas multisensor device deployed on the field in an Italian city. Hourly responses averages are recorded along with gas concentrations references from a certified analyzer. Multivariate (15) and time series Has missing values.

## Hints

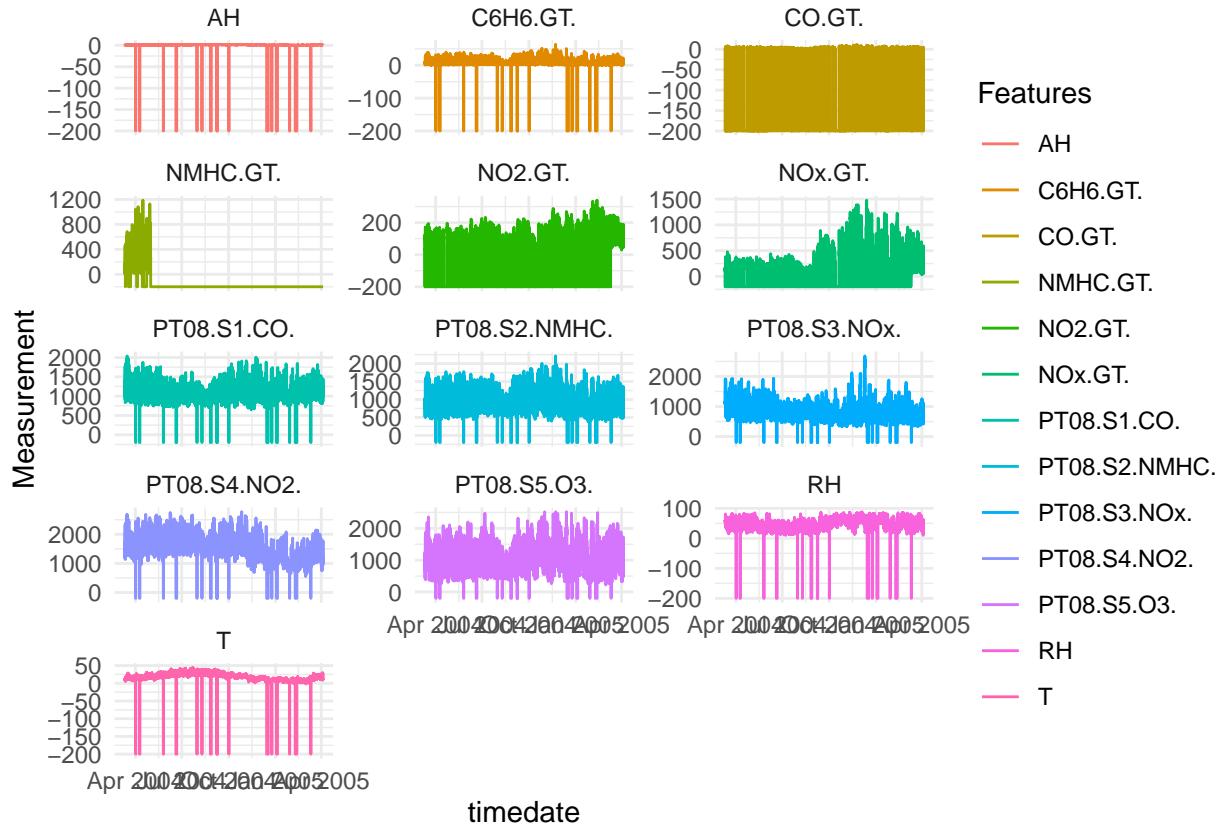
- See options of `read.table()` for correct import

## Part B: Import and Visualize

- Load the data and convert to tsibble.
  - Make sure dates and hours are converted into proper time objects
  - Remove incomplete days at beginning and end of data
- Plot the data as is, preferably as multiple panels in a single plot

- Describe the data. What is most striking?

```
airqual %>%
  pivot_longer(!timedate, names_to="Features", values_to="Measurement") %>%
  ggplot(aes(x = timedate, y = Measurement, col = Features)) +
  geom_line() +
  theme_minimal() +
  facet_wrap(~ Features, scales = "free_y", ncol = 3)
```



There seems to be multiple -200 in all the numerical (integer and Categorical) data that seems to be outliers or missing values. Should probably be removed or imputed from the dataset.

### Part C: PCA of data as is

- Perform PCA on the data as prepared in B

```
pc <- prcomp(airqual[,-1])
summary(pc)

## Importance of components:
##                               PC1        PC2        PC3        PC4        PC5        PC6
## Standard deviation    766.6076  396.3514  240.73023 172.96207 139.37119 87.72389
## Proportion of Variance 0.6736   0.1801   0.06643   0.03429   0.02226  0.00882
## Cumulative Proportion  0.6736   0.8537   0.92012   0.95441   0.97668  0.98550
##                               PC7        PC8        PC9        PC10       PC11       PC12
## Standard deviation    72.12259  66.31383  48.62607 23.26879 11.78504 2.29019
## Proportion of Variance 0.00596  0.00504  0.00271  0.00062  0.00016 0.00001
## Cumulative Proportion  0.99146  0.99650  0.99921 0.99983 0.99999 1.00000
```

```

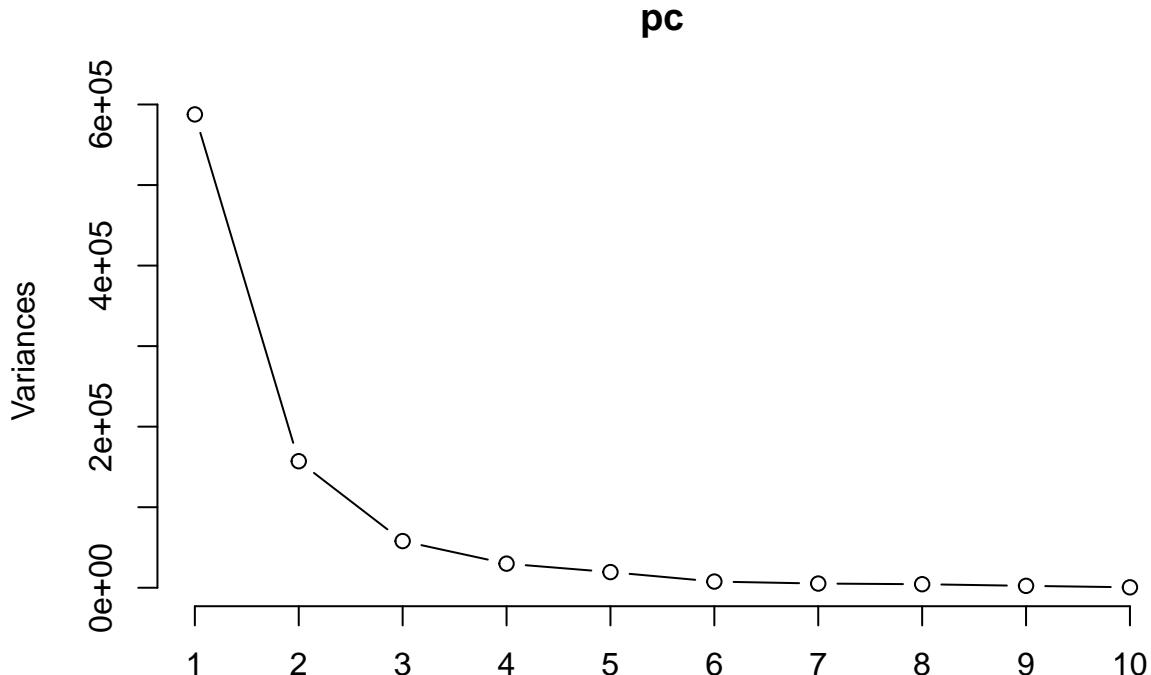
##                               PC13
## Standard deviation      0.7725
## Proportion of Variance 0.0000
## Cumulative Proportion  1.0000

```

- Create a screeplot and create biplots for 1st and 2nd and for 2nd and 3rd PCs

### Screeplot

```
plot(pc, type = "1")
```



```

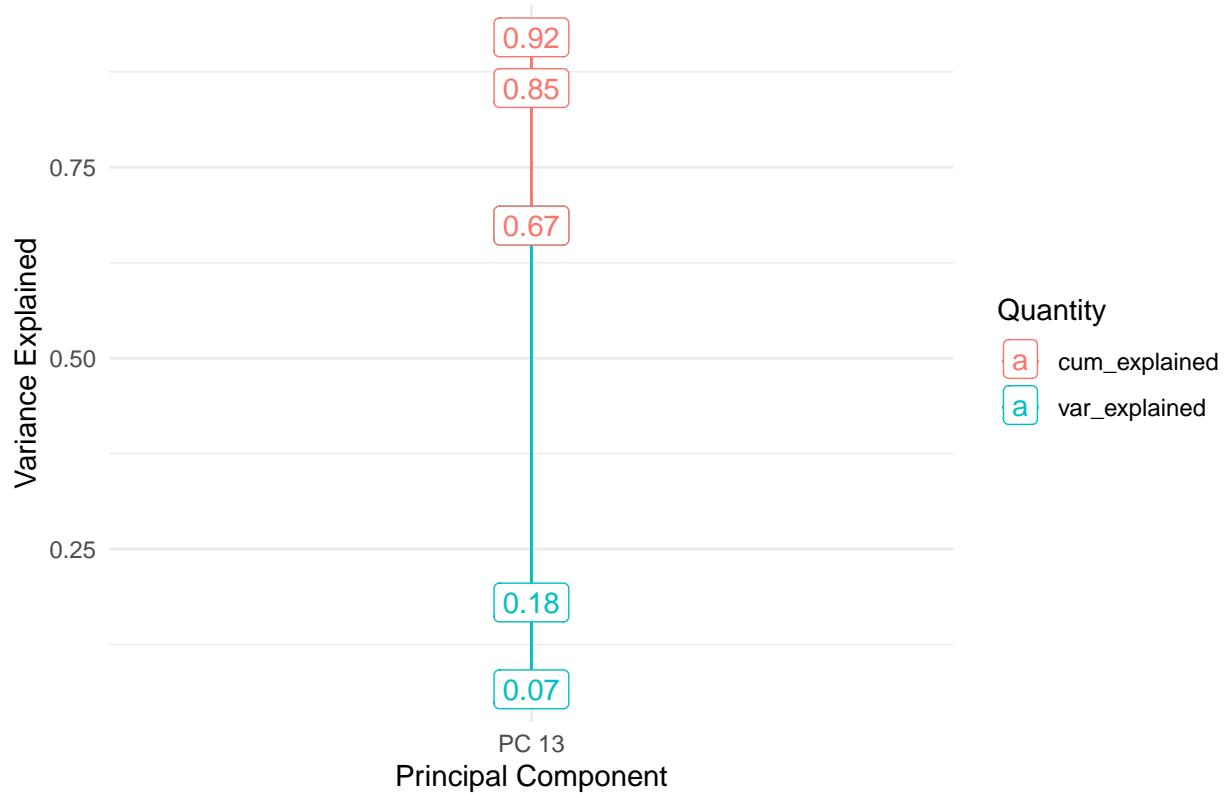
pc_v <- data.frame(PC = paste0("PC ", ncol(pc$x)),
                     var_explained = pc$sdev^2 / sum(pc$sdev^2)) %>%
  mutate(cum_explained = cumsum(var_explained))

pp <- pc_v[1:3, ] %>%
  pivot_longer(!PC, names_to="Quantity", values_to="Explained") %>%
  ggplot(aes(x = PC, y = Explained, color=Quantity, group=Quantity)) +
  geom_line() + geom_point() +
  theme_minimal() +
  labs(title = "Variance Explained", x = "Principal Component",
       y = "Variance Explained")

pp + geom_label(aes(label = round(Explained, 2)))

```

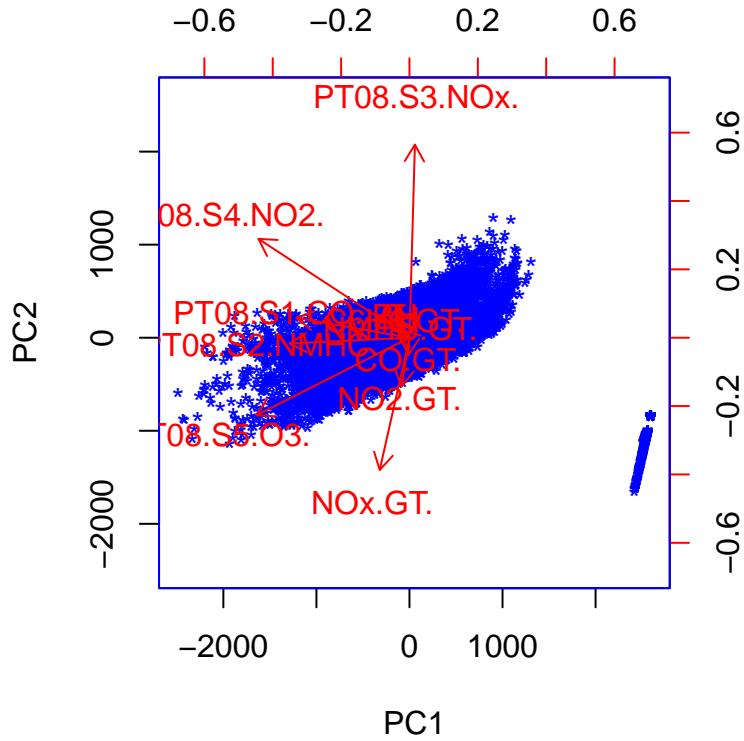
## Variance Explained



0.91 of variance is explained in the first 3 principal components

## Biplots

```
biplot(pc, scale=0, col=c('blue', 'red'), xlabs=rep('*', nrow(pc$x[, 1:3])))
```



- Can clearly see the -200 outliers
- PRO8.S3.NOx positively correlated between PC1 and PC2
- NOX.GT. opposite
- Plot the scores for the PCs
- Comment on the results. Can you relate some features to your observations in part B?

#### Score plot

```
d_pc <- data.frame(Time=airqual[,1], pc$x[,1:3])
head(d_pc)
```

```
##          timedate      PC1      PC2      PC3
## 1 2004-03-10 18:00:00 -530.23707 228.6828 177.0456
## 2 2004-03-10 19:00:00 -212.28933 377.7837 210.4454
## 3 2004-03-10 20:00:00 -309.51936 313.9656 247.2225
## 4 2004-03-10 21:00:00 -396.57641 230.7925 242.9392
## 5 2004-03-10 22:00:00 -192.91228 317.7539 298.5658
## 6 2004-03-10 23:00:00   27.54921 441.3118 357.9443
```

#### Hints

- `ggfortify` provides `autoplot()` for PCA results for ggplot-style biplots
- To plot the scores, you can use the same code as for plotting the original data

#### Part D: Missing values

- Identify missing values in the time series

The website says that all -200 are NA

- Investigate to which degree missing values occur at the same time for multiple sensors
- Is one or are multiple sensors behaving peculiarly? How would you handle this?

NHMC.GT has almost all missing values after a certain time, as seen in the visualization, therefore remove to not skew the result away from real data.

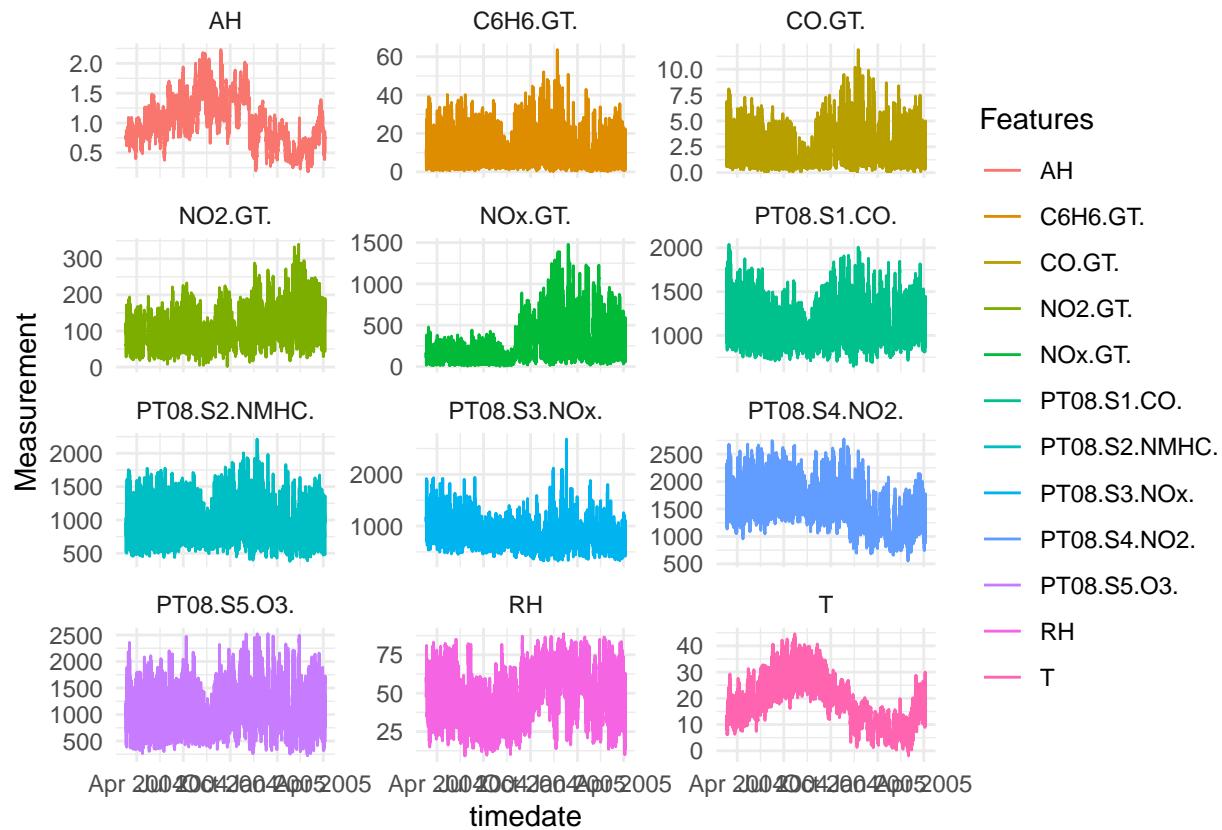
- Discuss options for handling missing values: (a) drop all time points containing any missing value, (b) impute values for missing values. In case of (b) choose a method for imputation. Justify your decisions.
- (a) By dropping all NA we only get real data to do analysis on, but we miss out on number of datapoints so our overall accuracy becomes lower. In addition it is not possible as we need a regular time axis.
- (b) By imputing we can still use rows that are mostly NA free while having the model be skewed as little as possible by these values.
  - Choose to set the NA to the mean value of the overall dataset, thereby making the datapoint the most probable if we were to get it randomly. By using rolling average we also get the most probable next step from the previous k steps, therefore making it more locally accurate than just using mean.
- At the end of this step, you should have a version of the data containing only valid values. Plot these data as in Part B.

```
airqual_c <- airqual %>%
  dplyr::select(-c("NMHC.GT."))

airqual_c[airqual_c == -200] = NA

#airqual_c <- imputeTS::na_mean(airqual_c, option = "mean")
# Missing Value Imputation by Weighted Moving Average, Simple moving average
# Plain rolling average
airqual_c <- imputeTS::na_ma(airqual_c, k = 4, weighting = "simple")

airqual_c %>%
  pivot_longer(!timedate, names_to="Features", values_to="Measurement") %>%
  ggplot(aes(x = timedate, y = Measurement, col = Features)) +
  geom_line() +
  theme_minimal() +
  facet_wrap(~ Features, scales = "free_y", ncol = 3)
```



### Hints

- Remember `imputeTS`
- You can apply its functions to an entire dataframe, will be done column-wise

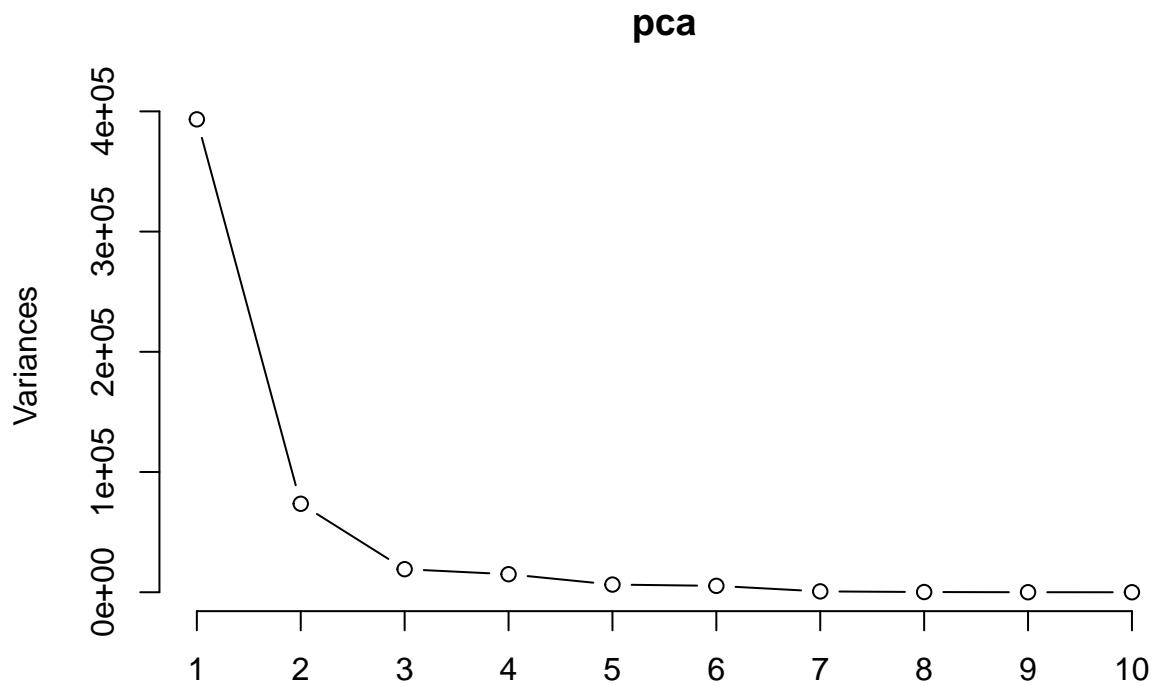
### Part E: PCA of cleaned data

- Perform PCA on the data as prepared in D

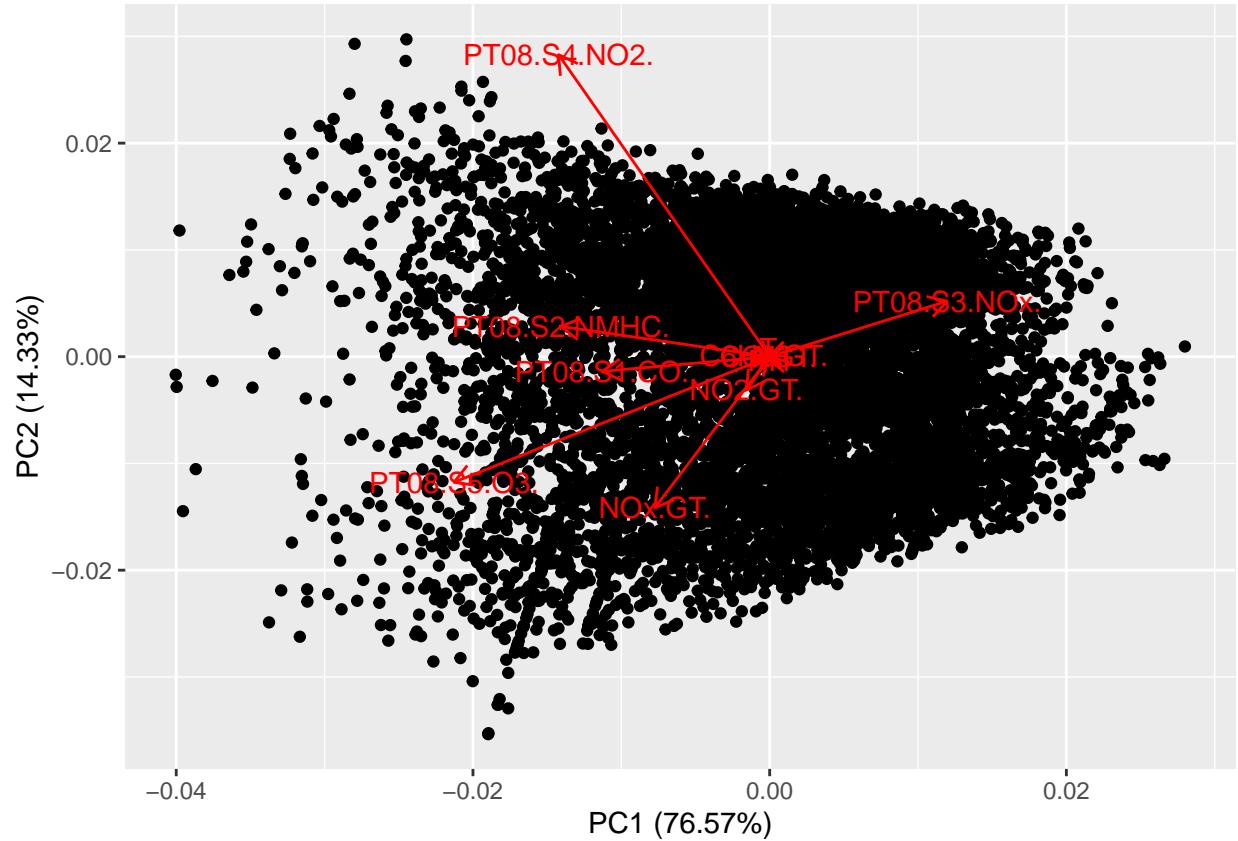
```
pca <- prcomp(airqual_c[, -1])
```

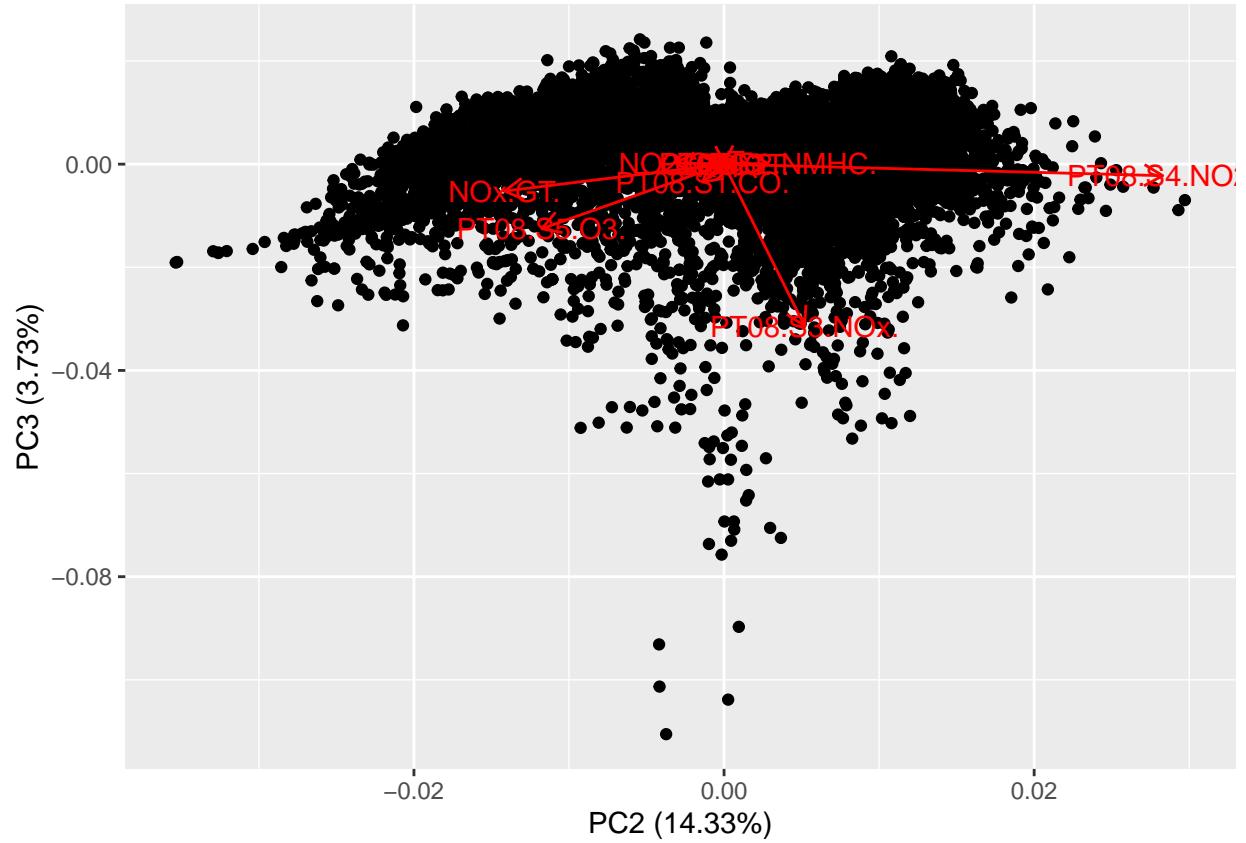
- Create a screeplot and biplots for 1st/2nd, 2nd/3rd, 3rd/4th PC

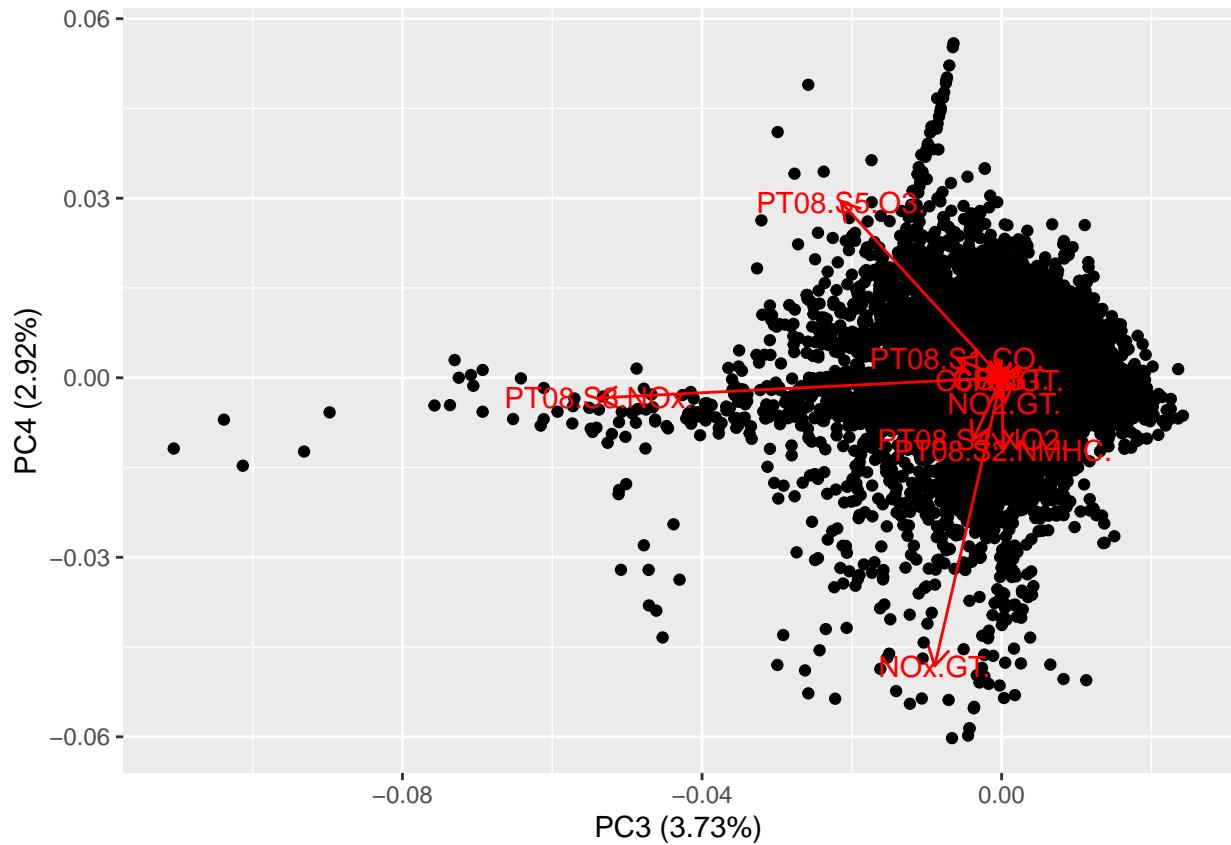
```
# screeplot
pca <- prcomp(airqual_c[, -1])
plot(pca, type = "1")
```



```
# biplots
autoplot(pca, x=1,y=2, loadings=TRUE, loadings.label=TRUE)
```







- Compute total variance explained by 1st, 1st and 2nd, 1st to 3rd, ... PCs

```
#Variance explained
summary(pca)$importance[3,]
```

```
##      PC1      PC2      PC3      PC4      PC5      PC6      PC7      PC8      PC9      PC10 
## 0.76570 0.90897 0.94631 0.97548 0.98791 0.99826 0.99964 0.99997 1.00000 1.00000 
##      PC11     PC12 
## 1.00000 1.00000
```

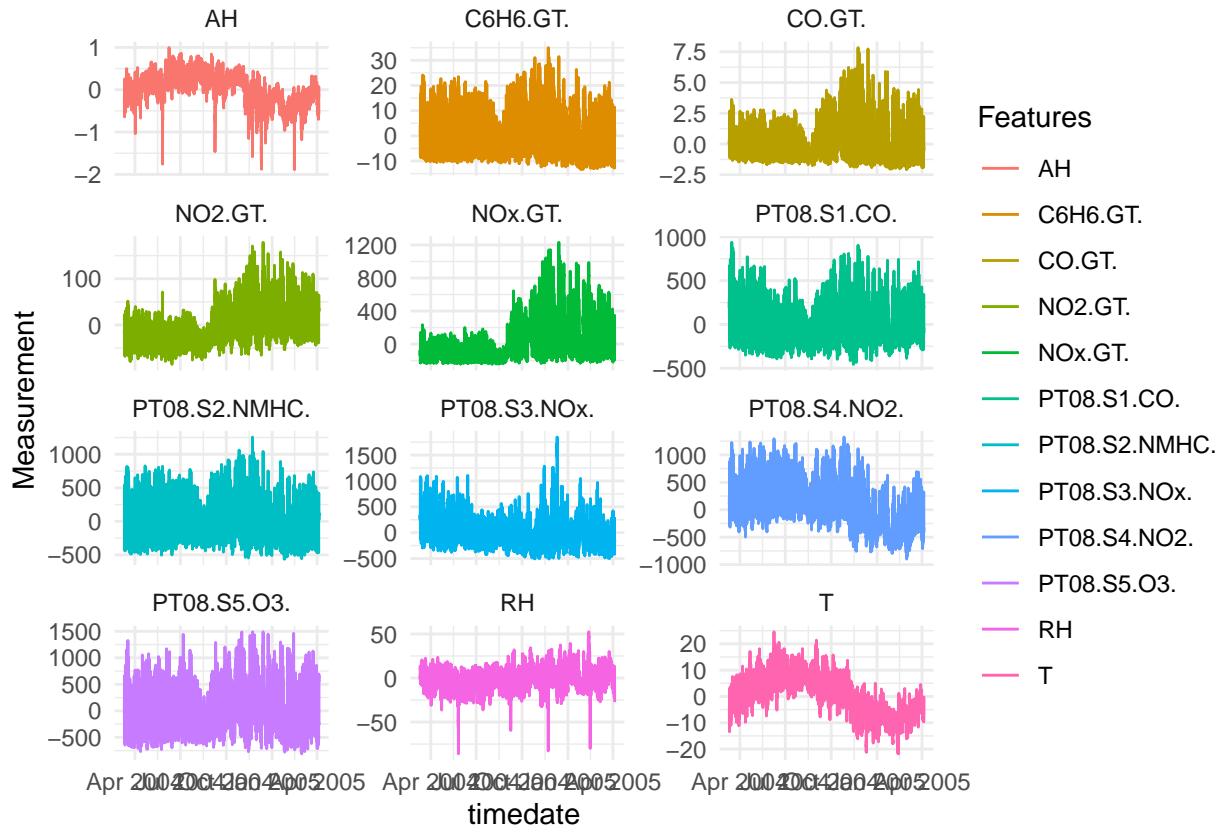
- Choose how many PCs to keep and transform data back to original sample space

Keep to 0.99 mark, so PC1 to PC6

```
t <- airqual_c$timedate
x_1 <- pca$x[, 1:6] %*% t(pca$rotation[, 1:6])
x_2 <- t(pca$center + pca$scale * t(x_1))
```

- Plot the result against the cleaned data, compare and discuss

```
data.frame(timedate = airqual_c$timedate, x_1) %>%
  pivot_longer(!timedate, names_to="Features", values_to="Measurement") %>%
  ggplot(aes(x = timedate, y = Measurement, col = Features)) +
  geom_line() +
  theme_minimal() +
  facet_wrap(~ Features, scales = "free_y", ncol = 3)
```

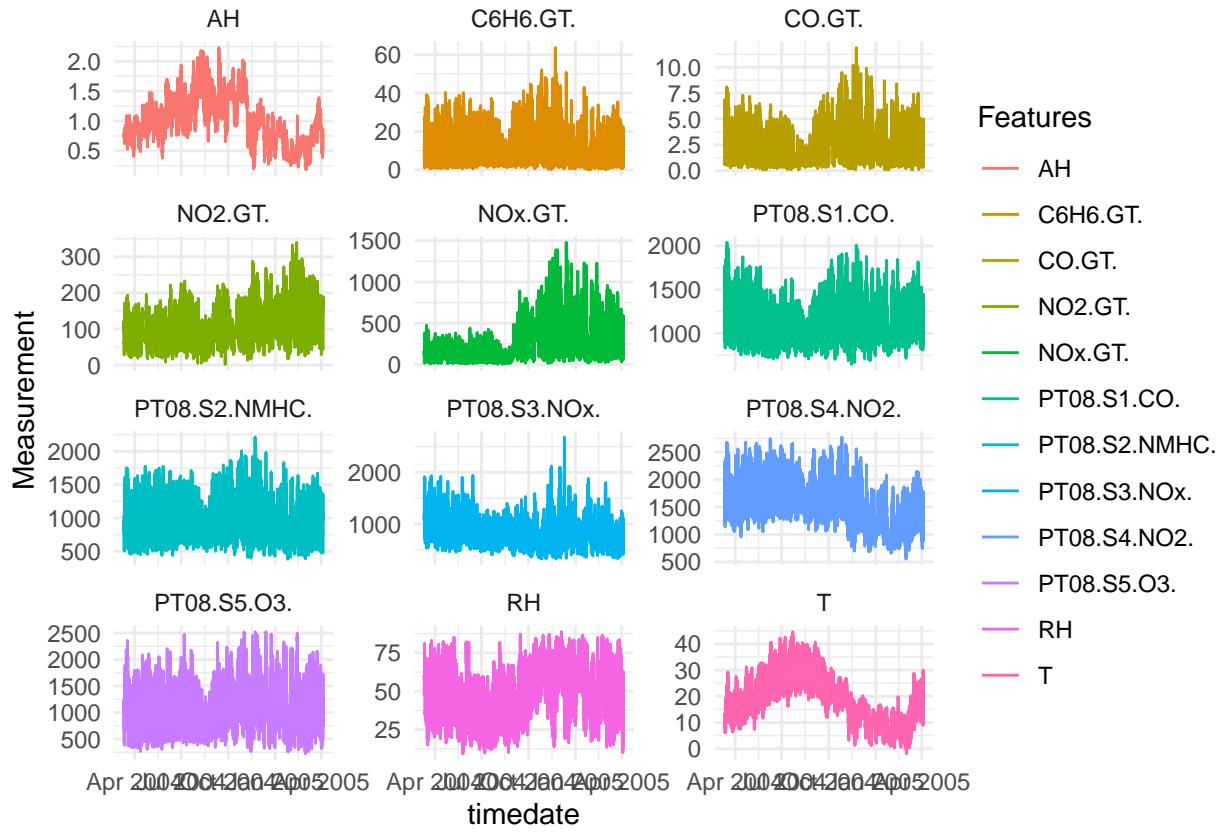


```

# data.frame(timedate = airqual_c$timedate, x_2) %>%
#   pivot_longer(!timedate, names_to="Features", values_to="Measurement") %>%
#   ggplot(aes(x = timedate, y = Measurement, col = Features)) +
#   geom_line() +
#   theme_minimal() +
#   facet_wrap(~ Features, scales = "free_y", ncol = 3)

airqual_c %>%
  pivot_longer(!timedate, names_to="Features", values_to="Measurement") %>%
  ggplot(aes(x = timedate, y = Measurement, col = Features)) +
  geom_line() +
  theme_minimal() +
  facet_wrap(~ Features, scales = "free_y", ncol = 3)

```

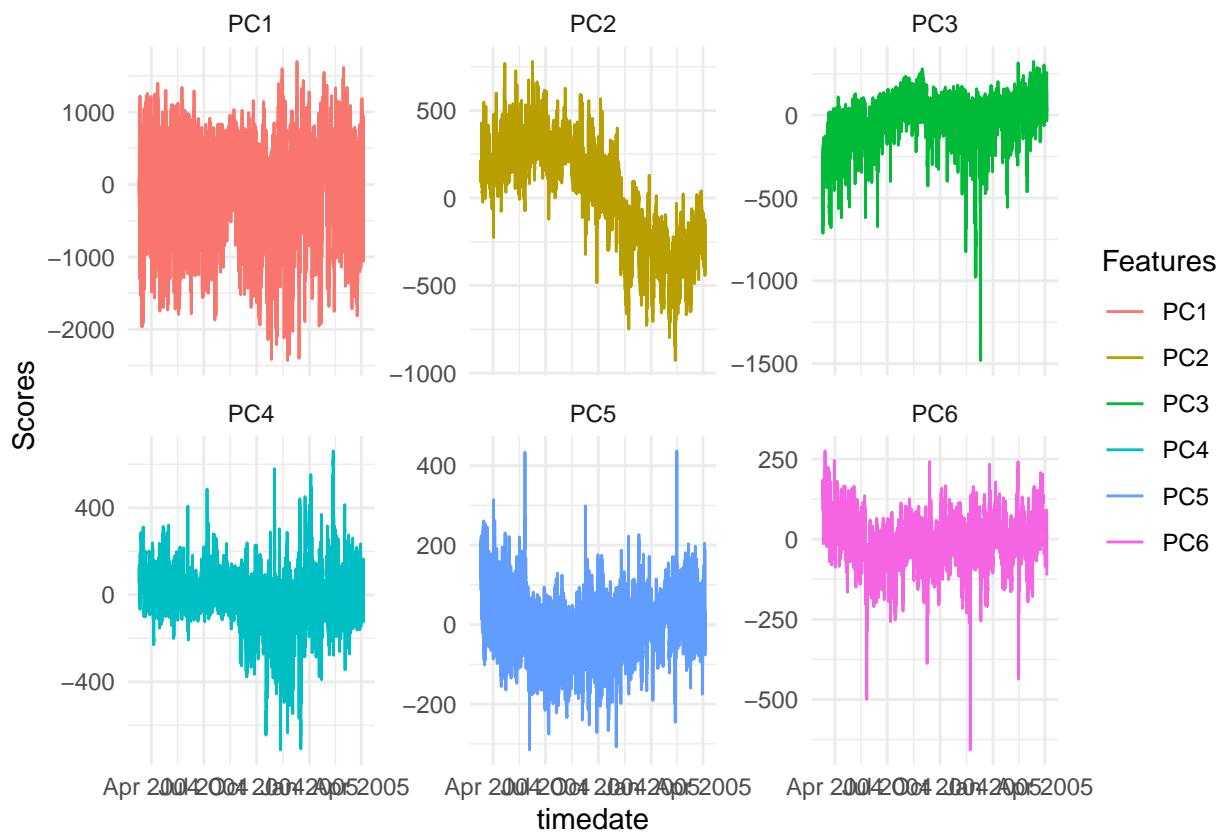


Looks mostly the same, but deviates slightly as we removed all after PC7.

- Also plot the scores, zoom in to short time intervals and look at periodicity

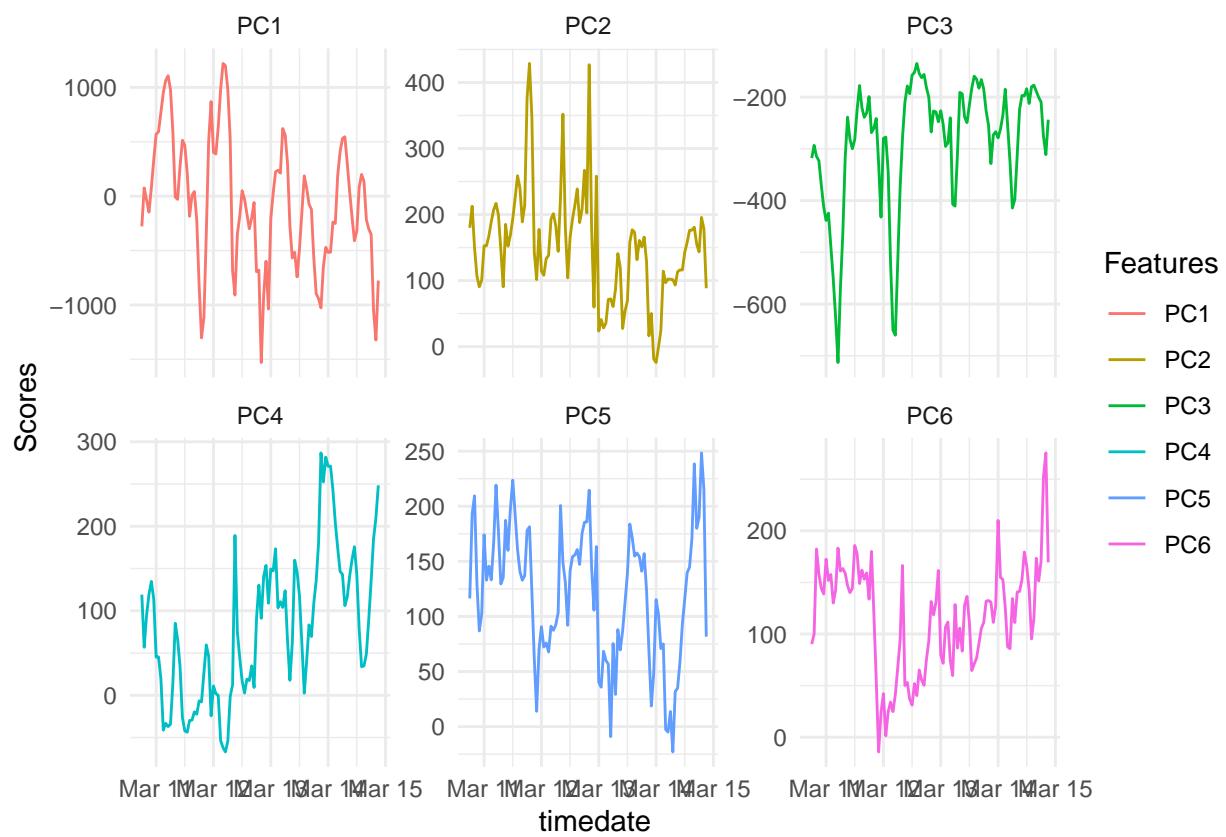
### Whole

```
data.frame(timedate = airqual_c$timedate, pca$x[,1:6]) %>%
  pivot_longer(!timedate, names_to="Features", values_to="Scores") %>%
  ggplot(aes(x = timedate, y = Scores, col = Features)) +
  geom_line() +
  theme_minimal() +
  facet_wrap(~ Features, scales = "free_y", ncol = 3)
```



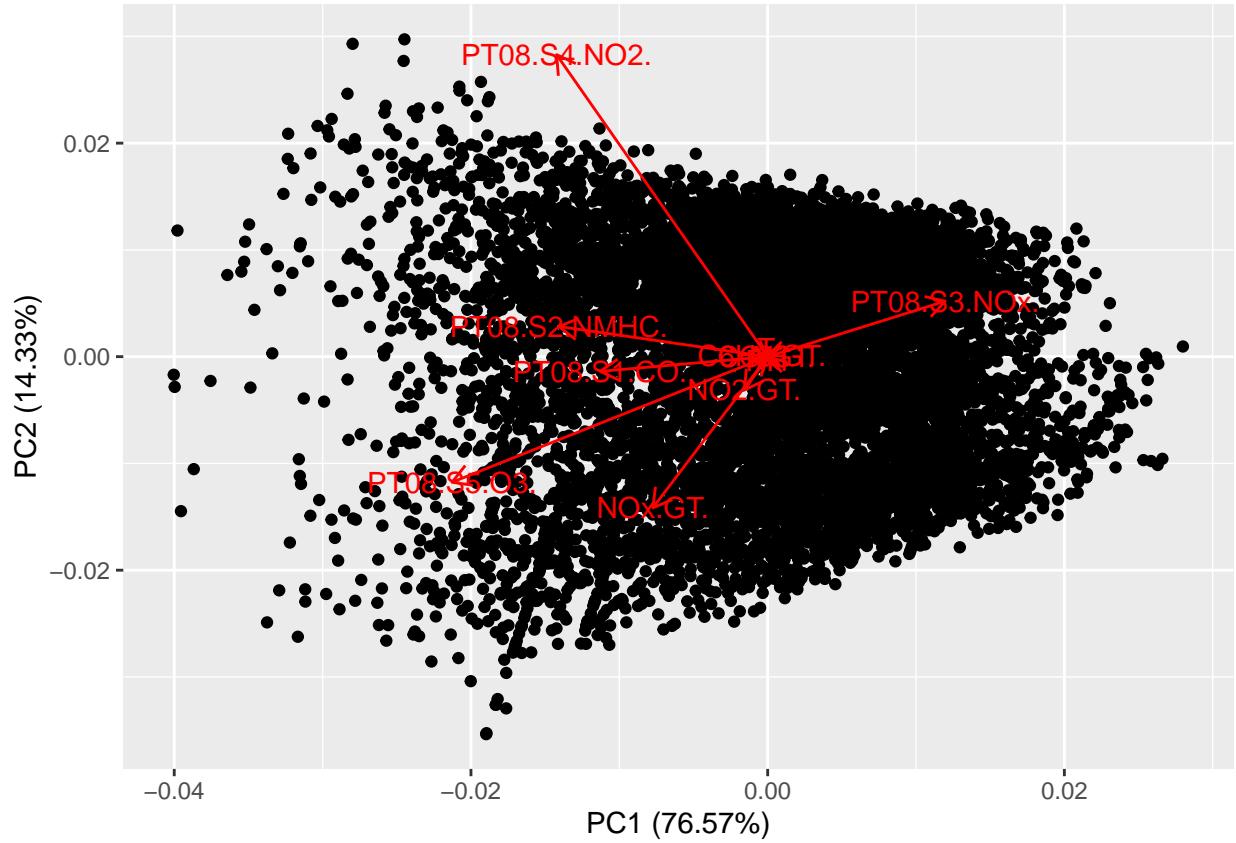
### Zoomed

```
data.frame(timedate = airqual_c$timedate[1:100], pca$x[1:100,1:6]) %>%
  pivot_longer(!timedate, names_to="Features", values_to="Scores") %>%
  ggplot(aes(x = timedate, y = Scores, col = Features)) +
  geom_line() +
  theme_minimal() +
  facet_wrap(~ Features, scales = "free_y", ncol = 3)
```



- Can you interpret certain PCs?

```
autoplot(pca, x=1,y=2, loadings=TRUE, loadings.label=TRUE)
```



```
data.frame(pca$rotation)
```

	PC1	PC2	PC3	PC4
## CO.GT.	-0.0018481988	-0.0007049452	-0.0014077536	-0.0043626668
## PT08.S1.CO.	-0.3276959483	-0.0400440912	-0.1009593770	0.0537630030
## C6H6.GT.	-0.0111497193	0.0023373054	-0.0045012384	-0.0078980451
## PT08.S2.NMHC.	-0.4085995243	0.0821947866	0.0037025890	-0.2073677556
## NOx.GT.	-0.2247579859	-0.4110090314	-0.1516711696	-0.8168812736
## PT08.S3.NOx.	0.3475027920	0.1508453517	-0.9111126863	-0.0570822819
## NO2.GT.	-0.0452313753	-0.0903548818	0.0065398011	-0.0728416677
## PT08.S4.NO2.	-0.4131515654	0.8212890426	-0.0647945566	-0.1760872468
## PT08.S5.O3.	-0.6176661288	-0.3415204901	-0.3633580682	0.4970953230
## T	-0.0021215778	0.0223286223	0.0146078597	-0.0068587449
## RH	-0.0019124088	-0.0116763960	-0.0125220967	-0.0031264697
## AH	-0.0001449796	0.0009679888	0.0006321034	-0.0003673498
	PC5	PC6	PC7	PC8
## CO.GT.	0.0002686689	0.000757493	-0.005358532	6.260908e-06
## PT08.S1.CO.	0.6259430252	0.695331376	0.014987171	-4.858268e-02
## C6H6.GT.	0.0163460121	-0.017403082	0.013341054	7.606726e-03
## PT08.S2.NMHC.	0.5577650907	-0.669313381	0.114878545	9.781349e-02
## NOx.GT.	-0.2518500677	0.132534035	0.073466372	-6.084502e-02
## PT08.S3.NOx.	0.1312507403	-0.072666427	-0.017993617	9.409716e-03
## NO2.GT.	0.0633409652	-0.041565772	-0.965138888	2.139533e-01
## PT08.S4.NO2.	-0.3082878182	0.135577313	-0.073581584	-8.415517e-03
## PT08.S5.O3.	-0.3340438218	-0.102865230	-0.001629221	-2.331928e-02
## T	-0.0070602392	-0.022954286	-0.040469919	-3.383485e-01

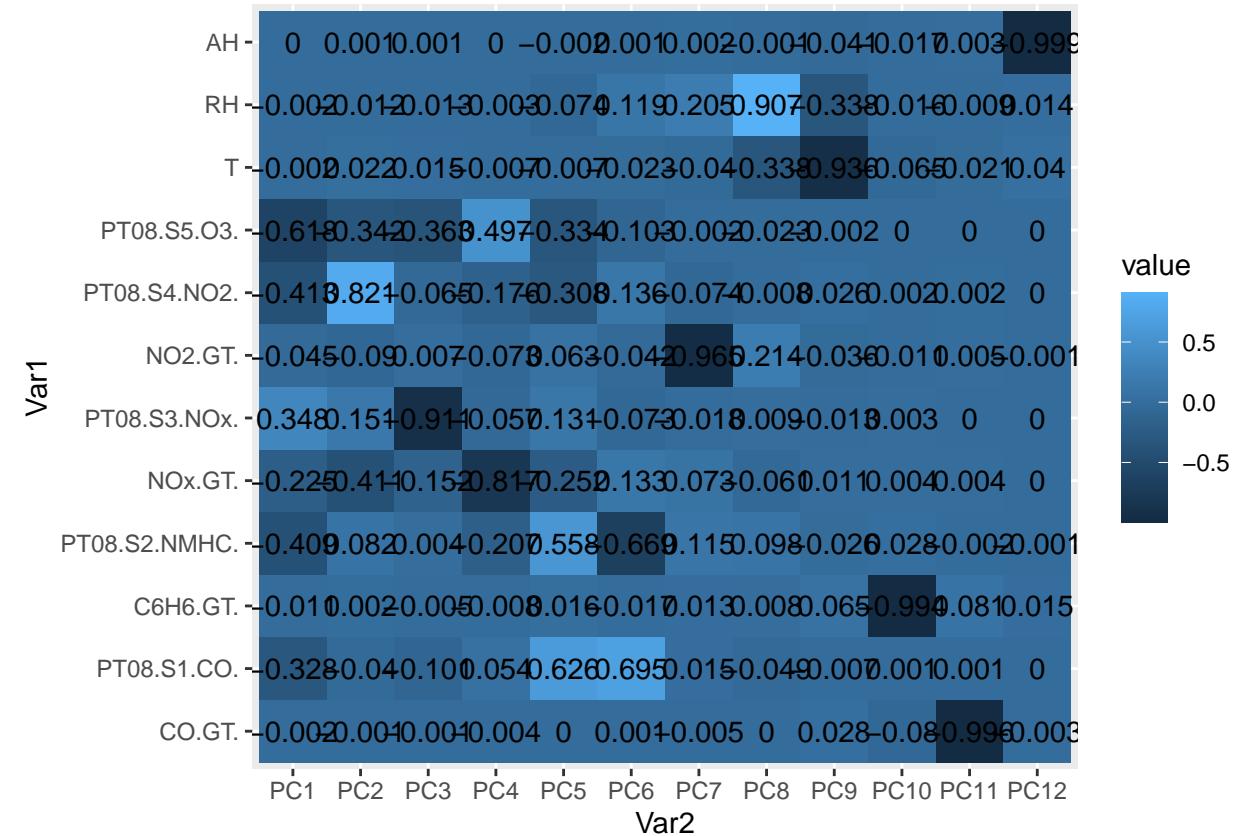
```

## RH          -0.0740118957  0.119022190  0.205178331  9.073875e-01
## AH          -0.0021484402  0.001182079  0.002142285 -1.130502e-03
##             PC9          PC10          PC11          PC12
## CO.GT.      0.028421694 -0.0798924563 -0.9963659489 -3.164062e-03
## PT08.S1.CO. -0.007152100  0.0011586314  0.0008637521 -2.323526e-04
## C6H6.GT.    0.065223128 -0.9939169941  0.0814893649  1.470071e-02
## PT08.S2.NMHC. -0.025579860  0.0277063380 -0.0023210316 -1.066060e-03
## NOx.GT.     0.011434225  0.0038849340  0.0041500425  2.374803e-04
## PT08.S3.NOx. -0.013083805  0.0034508345  0.0002142262 -3.947865e-04
## NO2.GT.     -0.035903855 -0.0106244017  0.0054654120 -8.600059e-04
## PT08.S4.NO2.  0.025888303  0.0016380067  0.0020691370  4.622803e-04
## PT08.S5.O3.  -0.002484763 -0.0002733566 -0.0004846717  7.147926e-05
## T            -0.936057264 -0.0653693331 -0.0213934536  4.010257e-02
## RH           -0.337718440 -0.0162504714 -0.0093587511  1.391709e-02
## AH           -0.041331962 -0.0172440281  0.0033651508 -9.989843e-01

melt(pca$rotation) %>%
  ggplot(aes(Var2, Var1)) +
  geom_tile(aes(fill = value)) +
  geom_text(aes(fill = value, label = round(value, 3)))

```

## Warning in geom\_text(aes(fill = value, label = round(value, 3))): Ignoring  
## unknown aesthetics: fill



By looking at the loadings of PC1 one can see that PT08.S5.O3 is the most important feature with respect to explainable variance.

## Task 2: STL and correlation on weather data

### Part A: Data collection for a single station

Based on material from the lectures, write an R function that can obtain a daily average temperature series for a meteorological station from the Norwegian Met Institute's Frost service. The function shall return a tibble.

```
.client_id <- str_trim(read_file("client_id.txt"))

# Server to collect data from and resource we want from server
server <- "frost.met.no"
resource <- "observations/v0.jsonld"

# Station(s) we want data for. SN17850 is the station ID for Ås (Blindern is SN18700)
sources <- 'SN17850'

# Type of data we want, P1D means daily data
elements <- 'mean(air_temperature P1D)'

# Time range we want data for
reference_time <- '1874-01-01/2023-12-31'

# Specify that we want mean temperature calculated from midnight to midnight
timeoffsets <- 'PTOH'

.query_url <- str_glue("https://.{client_id}@{server}/{resource}?sources={sources}&referencetime={reference_time}&elements={elements}&timeoffsets={timeoffsets}")

# Set this to TRUE to generate a json file
get_data_from_frost = FALSE
weather_file = "weather_data_json.rds.bz2"
if ( get_data_from_frost ) {
  raw_data <- try(fromJSON(URLencode(.query_url), flatten=TRUE))

  if ( class(raw_data) != 'try-error' ) {
    print("Data retrieved from frost.met.no!")
    write_rds(raw_data, weather_file, compress="bz2", text=TRUE) # JSON represents data as text
    print(str_glue("Raw data (JSON) written to '{weather_file}'"))
  } else {
    print("Error: the data retrieval was not successful!")
  }
} else {
  raw_data <- read_rds(weather_file)
  print(str_glue("Raw data (JSON) read from '{weather_file}'"))
}

## Raw data (JSON) read from 'weather_data_json.rds.bz2'
df <- unnest(raw_data$data, cols = c(observations))

head(df)

## # A tibble: 6 x 14
##   sourceId referenceTime      elementId value unit  timeOffset timeResolution
##   <chr>     <chr>          <chr>    <dbl> <chr> <chr>       <chr>
## 1 SN17850:0 1874-01-01T00:00:00~ mean(air~    3.2 degC  PTOH      P1D
```

```

## 2 SN17850:0 1874-01-02T00:00:00~ mean(air~ 4.1 degC PTOH P1D
## 3 SN17850:0 1874-01-03T00:00:00~ mean(air~ 2.5 degC PTOH P1D
## 4 SN17850:0 1874-01-04T00:00:00~ mean(air~ 0.3 degC PTOH P1D
## 5 SN17850:0 1874-01-05T00:00:00~ mean(air~ -3.9 degC PTOH P1D
## 6 SN17850:0 1874-01-06T00:00:00~ mean(air~ 1.9 degC PTOH P1D
## # i 7 more variables: timeSeriesId <int>, performanceCategory <chr>,
## # exposureCategory <chr>, qualityCode <int>, level.levelType <chr>,
## # level.unit <chr>, level.value <int>
df |> dplyr::select(referenceTime, value) |>
  mutate(referenceTime=as.Date(referenceTime)) |>
  rename(Date=referenceTime, Temp=value) |>
  as_tsibble() -> dc

```

## Using `Date` as index variable.

```
head(dc)
```

```

## # A tsibble: 6 x 2 [1D]
##   Date      Temp
##   <date>    <dbl>
## 1 1874-01-01  3.2
## 2 1874-01-02  4.1
## 3 1874-01-03  2.5
## 4 1874-01-04  0.3
## 5 1874-01-05 -3.9
## 6 1874-01-06  1.9

```

## Part B: Data preparation for a single station

- Identify gaps in the time series.

```
has_gaps(dc)
```

```

## # A tibble: 1 x 1
##   .gaps
##   <lgl>
## 1 TRUE

gaps <- count_gaps(dc)
head(gaps)

## # A tibble: 6 x 3
##   .from      .to       .n
##   <date>    <date>    <int>
## 1 1874-08-01 1874-08-31     31
## 2 1876-01-01 1877-07-31   578
## 3 1879-01-01 1879-01-31     31
## 4 1880-10-31 1880-10-31      1
## 5 1881-05-01 1881-05-31     31
## 6 1900-12-31 1900-12-31      1

```

- Assume that gaps up to 31 days are acceptable. Find the earliest date in the time series such that all following data have no gaps longer than 31 days. Limit the time series to this.

```
cutoff_date <- as.character(tail(gaps[gaps$.n >= 31,], n=1)$to)
cutoff_date
```

```

## [1] "1988-06-17"
identify that all dates after, even if 31 is included, is 1988-06-17
dc_cut <- dc %>% tsibble::filter_index(cutoff_date ~ .)
head(dc_cut)

## # A tsibble: 6 x 2 [1D]
##   Date      Temp
##   <date>    <dbl>
## 1 1988-06-18  18.7
## 2 1988-06-21  19.7
## 3 1988-06-22  18
## 4 1988-06-23  21.2
## 5 1988-06-25  21.3
## 6 1988-06-26  23

• Create a regular time series by filling gaps in the tsibble with n/a-s.

dc_cut_filled <- tsibble::fill_gaps(dc_cut, .full = TRUE)

head(dc_cut_filled)

## # A tsibble: 6 x 2 [1D]
##   Date      Temp
##   <date>    <dbl>
## 1 1988-06-18  18.7
## 2 1988-06-19  NA
## 3 1988-06-20  NA
## 4 1988-06-21  19.7
## 5 1988-06-22  18
## 6 1988-06-23  21.2

• Impute values for the n/a-s. Justify your choice of imputation method.

Choose to impute using simple rolling average, as you get the advantage of getting the most probable next step as well as being locally probable as opposed to using mean or median for the whole dataset.

dc_cut_filled_imp <- imputeTS::na_ma(dc_cut_filled, k = 4, weighting = "simple")

• You should now have a regular time series with only numeric values.

tsibble::has_gaps(dc_cut_filled_imp)

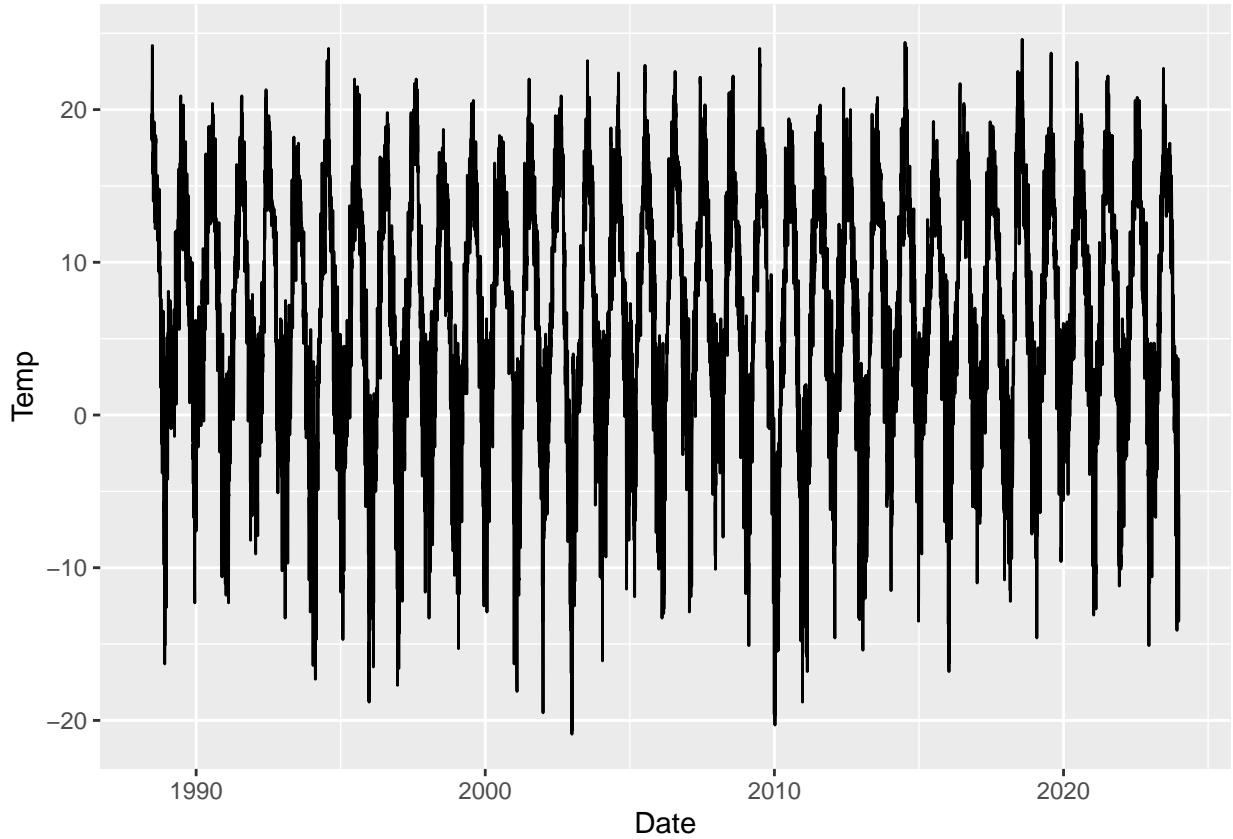
## # A tibble: 1 x 1
##   .gaps
##   <lgl>
## 1 FALSE

is.numeric(dc_cut_filled_imp$Temp)

## [1] TRUE

dc_cut_filled_imp %>%
  ggplot(aes(Date, Temp)) +
  geom_line()

```



- Remove all data for 29 February so all years have data for exactly 365 days.

```
dc_cut_filled_imp_noLeap <- dc_cut_filled_imp %>%
  dplyr::filter(!(month(Date) == 2 & day(Date) == 29))
head(dc_cut_filled_imp_noLeap)
```

```
## # A tsibble: 6 x 2 [1D]
##   Date      Temp
##   <date>    <dbl>
## 1 1988-06-18 18.7
## 2 1988-06-19 19.4
## 3 1988-06-20 19.4
## 4 1988-06-21 19.7
## 5 1988-06-22 18
## 6 1988-06-23 21.2
```

- Combine all this code into a function for re-use later. The function should receive the original tsibble from part A as input and return a new tsibble.

```
timeseries_cleaner <- function(table) {
  gaps <- count_gaps(table)
  cutoff_date <- as.character(tail(gaps[gaps$n >= 31,], n=1)$to)
  dc_cut <- dc %>% tsibble::filter_index(cutoff_date ~ .)
  dc_cut_filled_imp <- imputeTS::na_ma(dc_cut_filled, k = 4, weighting = "simple")
  dc_cut_filled_imp_noLeap <- dc_cut_filled_imp %>%
    dplyr::filter(!(month(Date) == 2 & day(Date) == 29))

  return(dc_cut_filled_imp_noLeap)
```

```

}

identical(timeseries_cleaner(dc), dc_cut_filled_imp_noLeap)

## [1] TRUE

```

### Hints

- tidyverse provides functions such as has\_gaps() and count\_gaps()

## Part C: Exploratory analysis for a single station

- Plot the temperature data as function of time
- Create density plots of original data and data with imputed values
- Turn the temperature data into a timeseries (ts) object
- Plot the autocorrelation function for lags up to 5.5 years; describe and discuss your observations
- Also plot the ACF only for short lags, up to four weeks
- Select some days distributed throughout the year and plot temperature as function of year for, e.g., 1 October, as a scatter plot. This plot can be useful to choose the seasonality window later (see Figs 7 and 8 in Cleveland et al, 1990)

## Part D: STL analysis

- Perform STL on the data. Explore different values for the seasonality and trend windows (remember that we want to look at trends over many years!), the choice between robust STL or not, and possibly the lowpass filter window. Describe your observations. It might be interesting to look at the ACF of the remainder in the STL result.
- Consult the original STL paper by Cleveland et al. (1990) for suggestions on how to choose STL parameters.
- Based on your analysis, can you suggest a set of STL parameters to use for further work?

## Part E: Multiple station analysis

- Obtain data from eight more stations. Two should be in the same part of Norway as the station from part A; then choose three stations each from two other parts of Norway. Data should cover several decades at least, so look for stations with long series.
- Preprocess the data as described in Part B. Find the latest starting date of any series and create a multivariate time series with data from all nine stations starting at this date.
- Obtain the cross-correlation matrix between the nine stations. Is there any structure in this 9x9 matrix?
- Perform STL individually on each of the nine stations using the parameters from part D. Compare the resulting trends. Are all STL results of equal quality?

### Hints

You can get a list of all available stations from Frost using

```
#.stations_url = str_glue("https://.client_id}@frost.met.no/sources/v0.jsonld")
#raw_stations <- fromJSON(URLencode(.stations_url), flatten=TRUE)
```

To limit this to stations with actual data, starting at least as early as 1950, coming from only some parts of Norway relevant columns, and limiting to relevant columns, filter the raw data as

```
# COUNTYS = c(fylke1, fylke2, etc) # replace with names of "fylker" you are interested in
#
# stations <- unnest(raw_stations$data, cols='id') />
#   select(id, validFrom, country, county, municipality, name, masl, `@type`) />
```

```
#   mutate(validFrom=as.Date(validFrom)) />
#   filter(`@type` == "SensorSystem" & validFrom <= "1950-01-01" & country == "Norge" & county %in% COUNTIES)
```

## Part F (bonus): PCA

- Perform PCA on the multivariate time series.