Individual assignment SBD

Hagen Aad Fock

ADS, 2021-2022

Contents

0. Prepare

▶ Load the R-packages you will use.

```
library(fpp3)
library(tseries)
library(expsmooth)

library(tidyverse)    # alternatively, this also loads %>%
library(knitr)
```

```r
library(mice) # for missing data imputation
library(VIM)
```

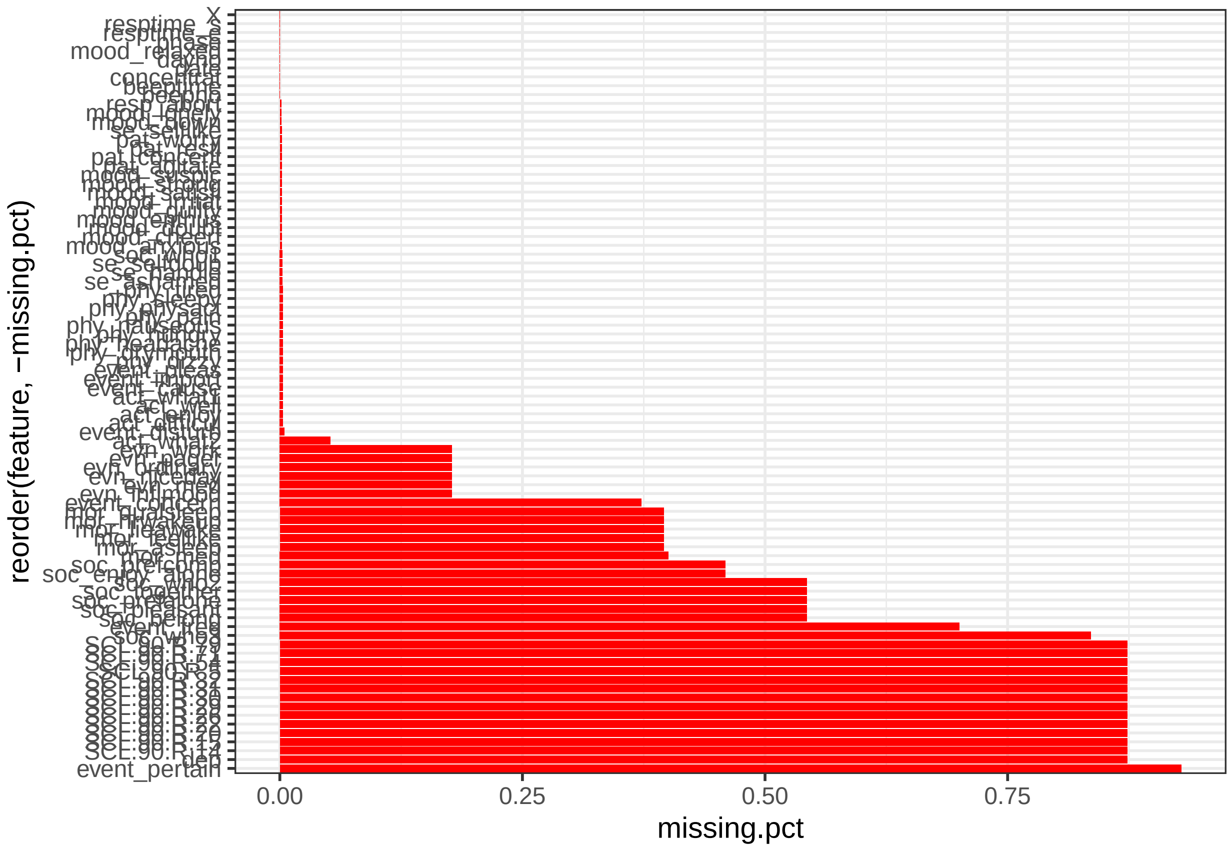
▶ Include R-code you used to load (and prepare) the data.

0.1 Checking for missing data

```r
set.seed(666)

data <- read.csv("../ESMdata/ESMdata.csv")
missing.values <- data %>%
  summarize_all(funs(sum(is.na(.))/n())) %>%
  gather(key="feature", value="missing.pct")
```

```
## Warning: `funs()` was deprecated in dplyr 0.8.0.
## Please use a list of either functions or lambdas:
##
##   # Simple named list:
##   list(mean = mean, median = median)
##
##   # Auto named with `tibble::lst()`:
##   tibble::lst(mean, median)
##
##   # Using lambdas
##   list(~ mean(., trim = .2), ~ median(., na.rm = TRUE))
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was generated.
```

```r
missing.values %>%
  ggplot(aes(x=reorder(feature,-missing.pct),y=missing.pct)) +
  geom_bar(stat="identity",fill="red")+
  coord_flip()+theme_bw()
```

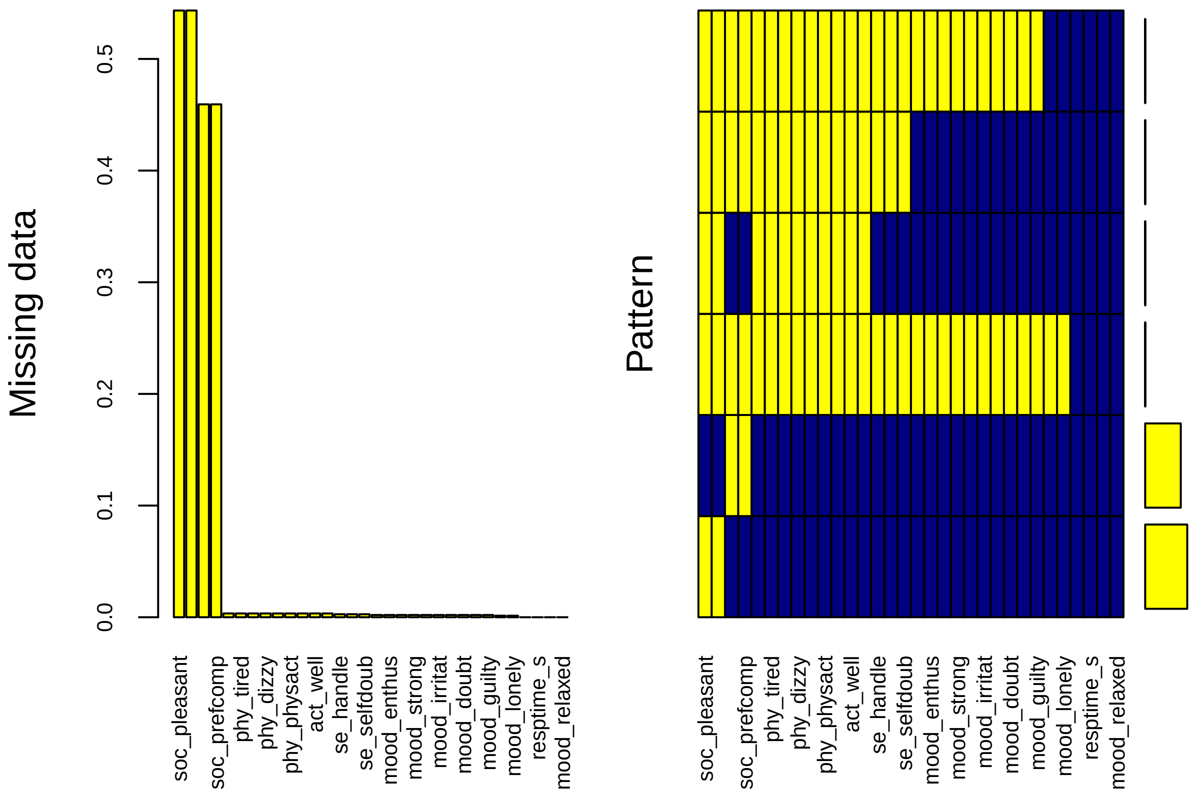


There are several variables that have a high percentage of missing data. Within my analysis I don't consider any variable that has a higher missing percentage then 3.4%.

```
relevant.variables <- c('date', 'resptime_s',  'concentrat',

                        'mood_relaxed', 'mood_satisfi', 'mood_enthus',
                        'mood_cheerf', 'mood_strong', 'se_selflike',
                        'se_handle',

                        'mood_irritat', 'mood_suspic', 'mood_doubt',
                        'se_ashamed', 'se_selfdoub',

                        'mood_down', 'mood_lonely', 'mood_anxious',
                        'mood_guilty',

                        'phy_hungry', 'phy_tired', 'phy_pain', 'phy_dizzy',
                        'phy_headache',

                        'soc_enjoy_alone', 'soc_prefcomp',
                        'soc_pleasant', 'soc_prefalone',

                        'phy_physact', 'act_difficul',
```

```r
                        'act_well', 'act_enjoy')
data <- data[,relevant.variables]

mice.plot <- aggr(data[,relevant.variables], col=c('navyblue','yellow'),
                    numbers=TRUE, sortVars=TRUE,
                    labels=names(relevant.variables), cex.axis=.7,
                    gap=3, ylab=c("Missing data","Pattern"))
```

```
## Warning in plot.aggr(res, ...): not enough horizontal space to display
## frequencies
```



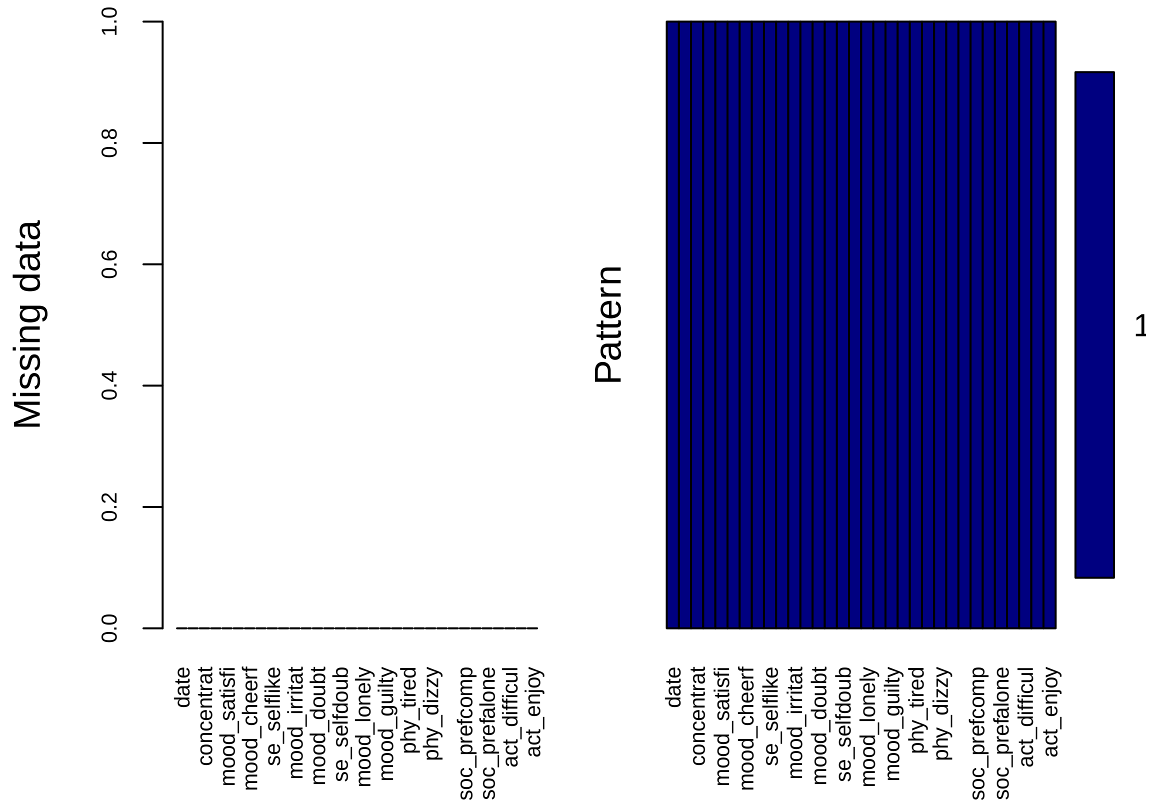

```
##
##  Variables sorted by number of missings:
##         Variable       Count
##      soc_pleasant 0.543360434
##     soc_prefalone 0.543360434
##   soc_enjoy_alone 0.459349593
##      soc_prefcomp 0.459349593
##        phy_hungry 0.003387534
##         phy_tired 0.003387534
##          phy_pain 0.003387534
##         phy_dizzy 0.003387534
##      phy_headache 0.003387534
##       phy_physact 0.003387534
```

```
##     act_difficul 0.003387534
##         act_well 0.003387534
##        act_enjoy 0.003387534
##        se_handle 0.002710027
##       se_ashamed 0.002710027
##      se_selfdoub 0.002710027
##     mood_satisfi 0.002032520
##      mood_enthus 0.002032520
##      mood_cheerf 0.002032520
##      mood_strong 0.002032520
##      se_selflike 0.002032520
##     mood_irritat 0.002032520
##      mood_suspic 0.002032520
##       mood_doubt 0.002032520
##     mood_anxious 0.002032520
##      mood_guilty 0.002032520
##        mood_down 0.001355014
##      mood_lonely 0.001355014
##             date 0.000000000
##       resptime_s 0.000000000
##       concentrat 0.000000000
##     mood_relaxed 0.000000000
```

```
# used this command to figure out the exact value of a variable
# missing.values[missing.values$feature == 'soc_prefcomp',]$missing.pct
```

```
## Warning: Number of logged events: 2
```

```
mice.plot <- aggr(data, col=c('navyblue','yellow'),
                    numbers=TRUE, sortVars=TRUE,
                    labels=names(data), cex.axis=.7,
                    gap=3, ylab=c("Missing data","Pattern"))
```




```
##
##  Variables sorted by number of missings:
##       Variable Count
##           date     0
##     resptime_s     0
##     concentrat     0
##   mood_relaxed     0
##   mood_satisfi     0
##    mood_enthus     0
##    mood_cheerf     0
##    mood_strong     0
##    se_selflike     0
##      se_handle     0
##   mood_irritat     0
##    mood_suspic     0
##     mood_doubt     0
##     se_ashamed     0
##    se_selfdoub     0
##      mood_down     0
##    mood_lonely     0
##   mood_anxious     0
##    mood_guilty     0
##     phy_hungry     0
```

```
##        phy_tired     0
##         phy_pain     0
##        phy_dizzy     0
##     phy_headache     0
##   soc_enjoy_alone    0
##     soc_prefcomp     0
##     soc_pleasant     0
##    soc_prefalone     0
##      phy_physact     0
##     act_difficul     0
##         act_well     0
##        act_enjoy     0
```

Now that every missing data was imputed we can make the tsibble. I create the tsibble by aggregating the mean of every day. The records of a day varied within a range of 1 and 10. I assumed that within such a day with many records the patients mood must have changed a lot otherwise they wouldn't record that much. Deriving from this thought I assumed that having the mean of such a day is more feasible for my analysis then including such a rollercoaster ride.

```
data$date <- dmy(data$date)
data <- aggregate(data[, 3:32], list(data$date), mean)
names(data)[1] <- 'date'

data <- data %>%
  as_tsibble(index = date)
```

1. General

▶ To be able to use fpp3, the data have to be a tsibble object. If they aren't already, transform them. Describe the structure of this object.

A tsibble is time series optimized tibble. It has in addition an Index that has an inherent ordering from past to present. Also has a key variable so multiple time series are possible. If there are implicit missing values they can be easily converted into explicit missing values with the `fill_gaps()` function. And around the tsibble is again a little tidyverse called `tidyverts` which includes a lot of libraries that are useful for time series analyses.

https://tsibble.tidyverts.org/

1.1. Describe your data

The data is about a man that was reducing his anti depression medication. Every variable that will be stated within this section was measured using a semi-random experience-sampling protocol. "The participant collected reports of momentary states up to 10 times a day over a period of 239 days.” https://doi.org/10.1159/000441458

“Depression is a mood disorder that causes a persistent feeling of sadness and loss of interest. Also called major depressive disorder or clinical depression, it affects how you feel, think and behave and can lead to a variety of emotional and physical problems. You may have trouble doing normal day-to-day activities, and sometimes you may feel as if life isn’t worth living.” https://www.mayoclinic.org/diseases-conditions/depression/symptoms-causes/syc-20356007

▶ What is your outcome variable; how was it measured (how many times, how frequently, etc.)?

The components of my outcome variable were measured at least everyday over 239 days and also sometimes sub daily. The total amount of measurements is approximately 1470 times.

Regarding the information from the article about depression I came up with the assumption that the best way to measure a depression is to observe the patients mood. Because there is such a variety of positive and negative moods I mixed them up in an overall variable called `depression_factor`.

All considered variables are within the following table.

name	description	scale	missing
`mood_relaxed`	I feel relaxed	+(1, 7)	0%
`mood_satisfi`	I feel satisfied	+(1, 7)	0.2%
`mood_enthus`	I feel enthusiastic	+(1, 7)	0.2%
`mood_cheerf`	I feel cheerful	+(1, 7)	0.2%
`mood_strong`	I feel strong	+(1, 7)	0.2%
`se_selflike`	I like myself	+(1, 7)	0.2%
`se_handle`	I can handle anything	+(1, 7)	0.3%
`positive_moods`	Accumulated positive moods. All variables with a ‘positive (1, 7)’ scale combined.	(-1, 1)	artificial
`mood_irritat`	I feel irritated	-(1, 7)	0.2%
`mood_suspic`	I feel suspicious	-(1, 7)	0.2%
`mood_doubt`	I feel indecisive	-(1, 7)	0.2%
`se_ashamed`	I am ashamed of myself	-(1, 7)	0.3%
`se_selfdoub`	I doubt myself	-(1, 7)	0.3%
`negative_moods`	Accumulated negative moods. All variables with a ‘negative (1, 7)’ scale combined.	(-1, 1)	artificial
`mood_down`	I feel down	-(-3, 3)	0.1%
`mood_lonely`	I feel lonely	-(-3, 3)	0.1%
`mood_anxious`	I feel anxious	-(-3, 3)	0.2%
`mood_guilty`	I feel guilty	-(-3, 3)	0.2%
`depressive_moods`	Accumulated depressive moods. All variables with a ‘negative (-3, 3)’ scale combined.	(-1, 1)	artificial

name	description	scale	missing
`depression_factor`	An accumulated factor that is an approximation to measure the depression. A combination of the variables `positive_moods`, `negative_moods` and `depressive_moods`	(-1, 1)	artificial

Because of the different scales within the variables, I had to transform them to the same base. I decided if the highest score on its scale is towards a positive or a negative mood for every variable. Then I scaled them depending on my assumption towards a scale of -1 and 1, where a positive score represents a positive mood, and a negative score represents a negative mood.

```
# the variable declaration is following
# _positive indicating if the highest value is a good mood
# _negative indicating if the highest value is a depressive mood
# _17       indicating if the scale is ( 1, 7)
# _33       indicating if the scale is (-3, 3)

# Data Normalization - Min-Max Normalization
# from (-3, 3) to (-1, 1)
Normalize <- function(atr, old_min, old_max)(
  return((atr - (old_min)) / (old_max - (old_min)) * (1 - (-1)) + (-1))
)

# For attributes on a scale (1, 7) where 1 indicates a negative mood and 7 a positive
NormalizePositive17 <-  function(atr) (
  return(Normalize((atr - 4), -3, 3))
)

# For attributes on a scale (1, 7) where 1 indicates a negative mood and 7 a positive
NormalizeNegative17 <- function(atr) (
  return(Normalize(((atr - 4) * -1), -3, 3))
)

# For attributes on a scale(-3, 3) where 3 is a depressive value.
NormalizeNegative33 <- function(atr) (
  return(Normalize((atr * -1), -3, 3))
)
```

```
# positive_moods
data$positive_moods <- data$mood_relaxed + data$mood_satisfi +
  data$mood_enthus + data$mood_cheerf + data$mood_strong +
  data$se_selflike + data$se_handle
```

```r
data$positive_moods <- Normalize(data$positive_moods,
                                  min(data$positive_moods),
                                  max(data$positive_moods))

# negative_moods
data$negative_moods <- data$mood_irritat + data$mood_suspic +
  data$mood_doubt + data$se_ashamed + data$se_selfdoub

data$negative_moods <- Normalize(data$negative_moods,
                                  min(data$negative_moods),
                                  max(data$negative_moods))

# depressive_moods
data$depressive_moods <- data$mood_down + data$mood_lonely +
  data$mood_anxious + data$mood_guilty

data$depressive_moods <- Normalize(data$depressive_moods,
                                  min(data$depressive_moods),
                                  max(data$depressive_moods))
```

After categorizing three different accumulated moods (`positive_moods`, `negative_moods`, `depressive_moods`) I accumulated and normalized each. Afterwards I added them together to the `depression_factor` and normalized the value again.

```r
data$depression_factor <- data$positive_moods + data$negative_moods +
  data$depressive_moods

data$depression_factor <- Normalize(data$depression_factor,
                                  min(data$depression_factor),
                                  max(data$depression_factor))
```

▶ What are the predictor variable(s) you will consider? Why would this make sense as a predictor?

I will chose all the variables that can be found in the next table.

name——— ——	description	scale
`phy_hungry`	I am hungry	-(1, 7)
`phy_tired`	I am tired	-(1, 7)
`phy_chanegable`	Physical conditions that can be changed. Variables `phy_hungry` and `phy_tired` combined.	(-1, 1)
`phy_pain`	I am in pain	-(1, 7)
`phy_dizzy`	I feel dizzy	-(1, 7)
`phy_headache`	I have a headache	-(1, 7)

name——— ——	description	scale
`phy_complain`	Physical conditions that can be described as complains. Variables `phy_pain`, `phy_dizzy` and `phy_headache` combined.	(-1, 1)
`phy_physact`	From the last beep on wards I was physically active	+(1, 7)
`soc_pleasant`	I find this company pleasant.	+(1, 7)
`soc_prefalone`	I prefer to be alone.	-(1, 7)
`soc_enjoy_alone`	I enjoy to be alone.	+(1, 7)
`soc_prefcomp`	I prefer being in company.	-(1, 7)
`soc_factor`	A factor that approximates the patients social needs. Variables `soc_pleasant`, `soc_prefalone`, `soc_enjoy_alone` and `soc_prefcomp` combined.	(-1, 1)

I assume that the physical variables are suitable to be used as predictor variables. I separated them into three variables the changeable physical conditions (`phy_chanegable`), the physical complains (`phy_complain`) and if the patient was physical active (`phy_physact`). `phy_chanegable` and `phy_complain` will be accumulated in the same manner as the artificial mood variables.

I think it makes sense to use `phy_chanegable` as a predictor because being tired or hungry are feelings that constantly influences the mood. Just as an example, the definition of the word 'hangry' is 'irritable or angry because of hunger'. https://www.merriam-webster.com/dictionary/hangry And having a bad sleep can trigger a bad mood.
The `phy_complain` variable makes also sense to be included as a predictor. Physical complains are obviously mood influential.
Doing sport (`phy_physact`) is also mood influential and thus also suitable to be used as predictor.

I also assume the social part (`soc_factor`) is a crucial factor to predict the mood. Therefore, it is added to be predictor variables.

```
# phy_chanegable
data$phy_chanegable <- data$phy_hungry + data$phy_tired

data$phy_chanegable <- Normalize(data$phy_chanegable,
                                 min(data$phy_chanegable),
                                 max(data$phy_chanegable))

# phy_complain
data$phy_complain <- data$phy_pain + data$phy_dizzy + data$phy_headache

data$phy_complain <- Normalize(data$phy_complain,
                               min(data$phy_complain),
                               max(data$phy_complain))
```

```r
# phy_physact
data$phy_physact <- NormalizePositive17(data$phy_physact)

# soc_factor
data$soc_pleasant <- NormalizePositive17(data$soc_pleasant)
data$soc_prefalone <- NormalizeNegative17(data$soc_prefalone)
data$soc_enjoy_alone <- NormalizePositive17(data$soc_enjoy_alone)
data$soc_prefcomp <- NormalizeNegative17(data$soc_prefcomp)

data$soc_factor <- data$soc_pleasant + data$soc_prefalone +
  data$soc_enjoy_alone + data$soc_prefcomp

data$soc_factor <- Normalize(data$soc_factor,
                             min(data$soc_factor),
                             max(data$soc_factor))
```

▶ What are the cause(s) you will consider? Why would this make sense as a cause?

I assume that a human being is complex but an encapsulated system on its own. But the surrounding factors influence whatever happens within a person. So I assume all activities with the environment like meeting people or doing something actively must be the cause of how the human being feels and also everything a person takes to him influences the internal behavior.

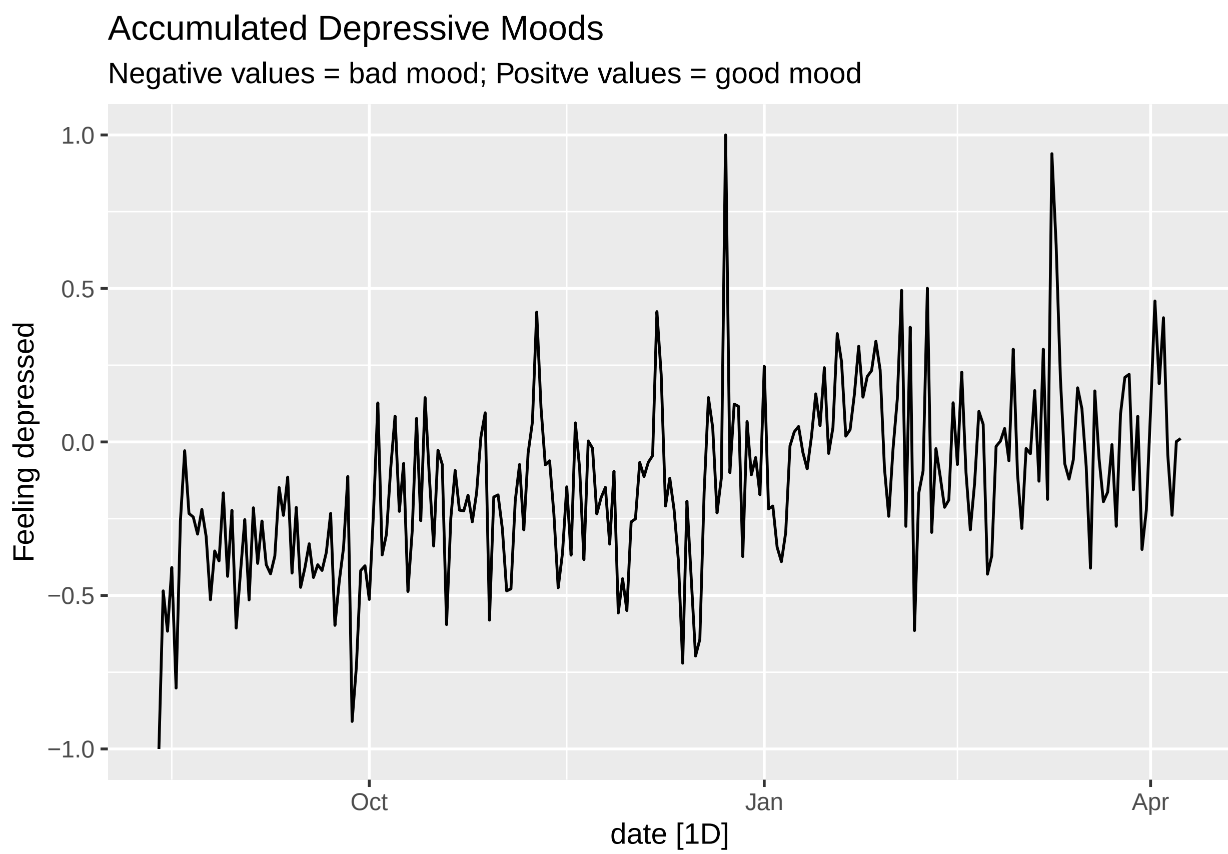
name	description	scale	missing
`act_difficul`	This (activity) requires effort	-(1, 7)	3.4%
`act_well`	I am good at this	+(1, 7)	3.4%
`act_enjoy`	I like doing this	+(1, 7)	3.4%
`act_complexity`	The complexity of a task. Variables `act_difficul`, `act_well` and `act_enjoy`combined.	(-1, 1)	artificial

```r
# act_complexity
data$act_difficul <- NormalizeNegative17(data$act_difficul)
data$act_well <- NormalizePositive17(data$act_well)
data$act_enjoy <- NormalizePositive17(data$act_enjoy)

data$act_complexity <- data$act_difficul + data$act_well + data$act_enjoy

data$act_complexity <- Normalize(data$act_complexity,
                                 min(data$act_complexity),
                                 max(data$act_complexity))
```

1.2. Visualize your data

▶ Create a sequence plot of the data with the function autoplot(). Interpret the results.

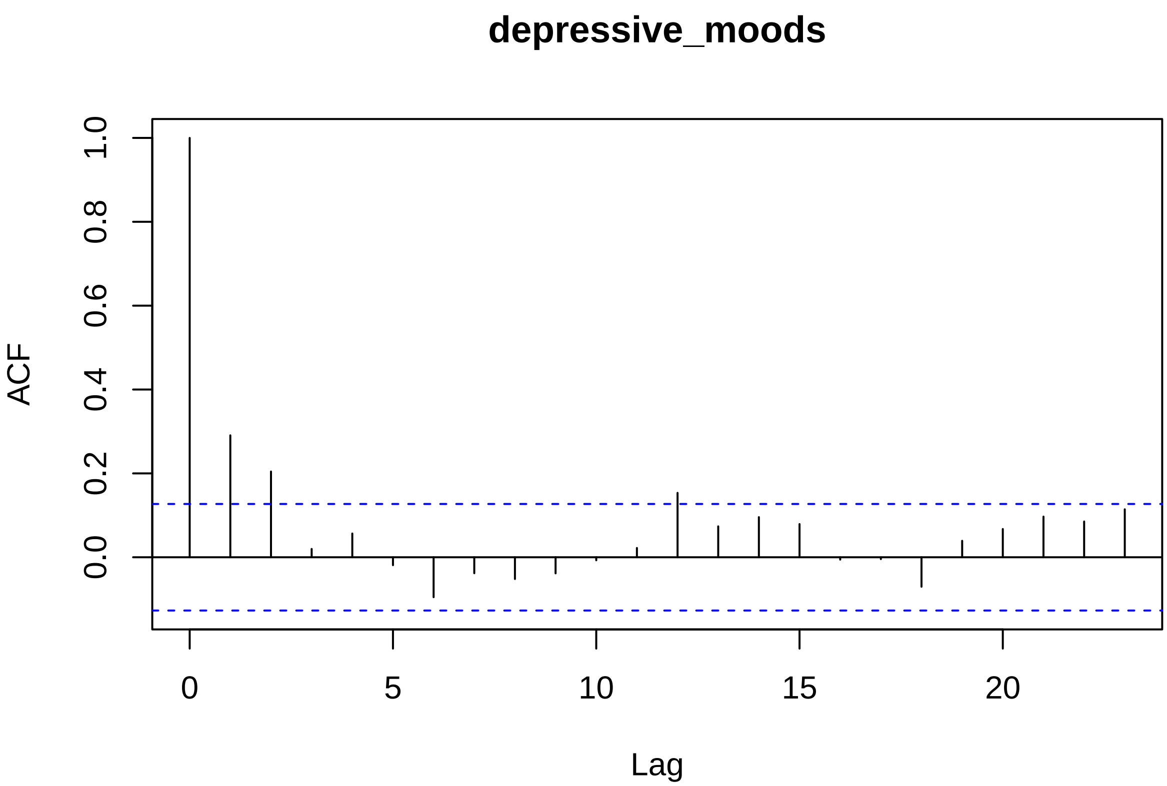
```
autoplot(data, depression_factor) +
  labs(title = "Accumulated Depressive Moods",
       subtitle = "Negative values = bad mood; Positve values = good mood",
       y = "Feeling depressed")
```



- We can see that there is almost always a fluctuation between feeling and not feeling depressed. It seems like there are few or no existing stable phases.
- On almost every good day (feeling not depressed) a bad day (feeling depressed) is the follow up. the most extreme situations are the October, just before the January

▶ Plot the autocorrelation function with the function acf(). Interpret the results.

```
acf(data[34])
```



▶ Based on (basic) content knowledge about the variable, and these visualizations, is there reason to assume the data are non-stationary and/or that there is a seasonal component?

2. Forecasting

2.1. SARIMA modeling

▶ Perform the Dickey-Fuller test. What is your conclusion?

```
adf.test(data$depression_factor)
```

```
## Warning in adf.test(data$depression_factor): p-value smaller than printed p-
## value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  data$depression_factor
## Dickey-Fuller = -5.6175, Lag order = 6, p-value = 0.01
## alternative hypothesis: stationary
```

▶ Fit an (S)ARIMA model to the data; what is the order of the model that was selected?

```r
data <- tsibble::fill_gaps(data)
fit_data <- model(data, ARIMA(depression_factor))
report(fit_data)
```

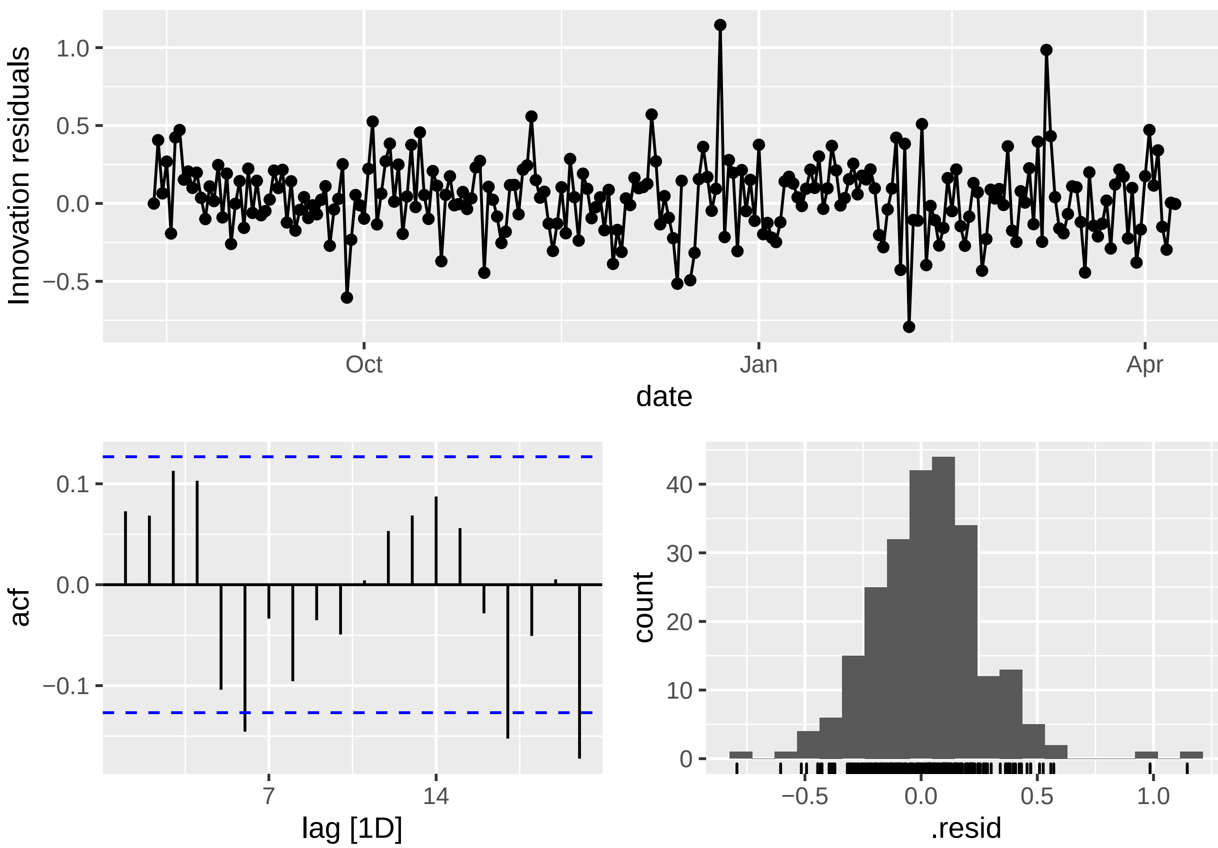
```
## Series: depression_factor
## Model: ARIMA(0,1,2)(1,0,0)[7]
##
## Coefficients:
##           ma1      ma2     sar1
##       -0.7557  -0.1538  -0.0886
## s.e.   0.0601   0.0603   0.0683
##
## sigma^2 estimated as 0.05958:  log likelihood=-2
## AIC=12.01   AICc=12.18   BIC=25.89
```

▶ Check the residuals of the model using the function gg_tsresiduals(). What is your conclusion?

```r
gg_tsresiduals(fit_data)
```

```
## Warning: Removed 1 rows containing missing values (geom_point).
```

```
## Warning: Removed 1 rows containing non-finite values (stat_bin).
```



```
augment(fit_data)
```

```
## # A tsibble: 239 x 6 [1D]
## # Key:       .model [1]
##    .model                   date       depression_fact~ .fitted   .resid   .innov
##    <chr>                    <date>                <dbl>   <dbl>    <dbl>    <dbl>
##  1 ARIMA(depression_facto~ 2012-08-13          -1        -0.999 -0.00100 -0.00100
##  2 ARIMA(depression_facto~ 2012-08-14          -0.485    -0.891  0.406    0.406
##  3 ARIMA(depression_facto~ 2012-08-15          -0.616    -0.681  0.0646   0.0646
##  4 ARIMA(depression_facto~ 2012-08-16          -0.409    -0.677  0.268    0.268
##  5 ARIMA(depression_facto~ 2012-08-17          -0.802    -0.608 -0.193   -0.193
##  6 ARIMA(depression_facto~ 2012-08-18          -0.259    -0.682  0.424    0.424
##  7 ARIMA(depression_facto~ 2012-08-19          -0.0285   -0.500  0.471    0.471
##  8 ARIMA(depression_facto~ 2012-08-20          -0.233    -0.385  0.152    0.152
##  9 ARIMA(depression_facto~ 2012-08-21          -0.244    -0.449  0.205    0.205
## 10 ARIMA(depression_facto~ 2012-08-22          -0.300    -0.400  0.0996   0.0996
## # ... with 229 more rows
```

2.2. Dynamic regression

▶ Include the predictor in an dynamic regression model (i.e., allow for (S)ARIMA residuals); what is the effect of the predictor?

▶ What order is the (S)ARIMA model for the residuals?

▶ Check the residuals of the model using the function gg_tsresiduals(). What is your conclusion?

2.3. Forecasts

▶ Choose a forecasting horizon, and indicate why this is a reasonable and interesting horizon to consider.

▶ Create forecasts based on the model without the predictor and plot these.

▶ Create forecasts based on the model with the predictor and plot these.

▶ Compare the plots of both forecasts (visually), and discuss how they are similar and/or different.

3. Causal Modeling

▶ Formulate a causal research question(s) involving the time series variable(s) you have measured.

▶ Which method we learned about in class (Granger causal approaches, interrupted time series, synthetic controls) is most appropriate to answer your research question using the data you have available? Why?

3.2 Analysis

Depending on the choice you made above, follow the questions outlined in 3.2a, 3.2b or 3.2c. If you chose a Granger causal analysis, it is sufficient to assess Granger causality in one direction only: you may evaluate a reciprocal causal relationship, but then answer each question below for both models.

3.2a Granger Causal analysis

▶ Visualize your putative cause variable(s) X and outcome variables Y.

▶ Train an appropriate ARIMA model on your outcome variable(s) Y, ignoring the putative cause variable(s) (X) but including, if appropriate, any additional covariates. If using the same model as fit in part 2, briefly describe that model again here.

▶ Justify what range of lags to consider for the lagged predictor(s). Use the CCF, but you may also justify this based on domain knowledge or substantive theory.

▶ Investigate whether adding your lagged "cause'' variables (X) improve the prediction of your effect variable(s) Y. Use model selection based on information criteria. Describe your final chosen model

3.2b Interrupted Time Series analysis

▶ Partition your dataset into pre- and post- intervention time periods and visualize this. Describe what the intervention is and when it takes place

▶ Train an appropriate ARIMA model on pre-intervention data of your outcome variable Y. If using the same model as fit in part 2, briefly describe that model again here.

▶ Use this model to create forecasts for the post-intervention time period. Visualize your forecasts (both point predictions and intervals) and the observed post-intervention data in a single plot.

▶ Compare your forecasts (both point predictions and intervals) to the observed post-intervention data. Describe if and how these differ from one another.

3.2c Synthetic Control Analysis

▶ Partition your dataset into pre- and post- intervention time periods. Describe what the intervention is and when it takes place. Describe your control series. Visualize your original time-series, control series, and the intervention period.

▶ Train an appropriate model on pre-intervention data of your outcome variable and the control series. Describe the model. Note: if using `CausalImpact`, describe the fitted model before visualizing the forecasts.

▶ Use this model to create forecasts for the post-intervention time period. Visualize your forecasts (both point predictions and intervals) and the observed post-intervention data in a single plot.

▶ Compare your forecasts (both point predictions and intervals) to the observed post-intervention data. Describe if and how these differ from one another.

3.3 Conclusion and critical reflection

▶ Based on the result of your analysis, how would you answer your causal research question?

▶ Making causal conclusions on the basis of your analysis is reliant on a number of assumptions. Pick a single assumption that is necessary in the approach you chose. Discuss the plausability and possible threats to the validity of this assumption in your specific setting (< 75 words)
