Rationale

Every year we start pre-growing some plants from seed indoors and eagerly wait for the moment we can finally transplant them to the outdoors. However, we want to make sure, that our precious seedlings are safe from frosty nights. As we recently moved to a new place (Germany), we checked with the locals what the best time for transplanting seedlings would be and heard about an old weather proverb (*Bauernregel* in German) which says that you should wait until the day of the Cold Sophia has passed. That day is the last of the so-called Ice Saints (or *Eisheiligen* in German), which are usually three to five (depending on where you live and whom you ask) name days of Saints in mid-May. The folklore has it, that once these days have passed, the nights will be guaranteed to be free from frost for the remaining growing season.

While perennial bushes like gorse can tolerate a sudden period of frost, most small seedlings couldn't.

So, when are these days of the Ice Saints? The answer is, as so often, interestingly not that easy. In most cases, people will tell you that the Ice Saints cover the period from May 12 to May 15, starting with the name day of St. Pancras, then St. Servatus and St. Boniface of Tarsus, and finally ending with the aforementioned St. Sophia. However, according to Wikipedia, "in 1582, the replacement of the Julian calendar by the Gregorian calendar involved omitting 10 days in the calendar. So if the folklore predates the calendar change, then the equivalent dates from the climatic point of view would be May 22–25." Which is very interesting indeed.

But since I don't want to leave the fate of my precious seedlings to the musings of long-gone Saints alone, let's employ the power of data and R to check if the Saints were right all along!

For that, I will

- download publicly available, historic weather data from the German Meteorological Service (Deutscher Wetterdienst, DWD),
- then import these into R,
- fit a logit model to the data using nested data frames,
- and finally make some predictions based on the model results about when it would be safe to plant the seedlings outside.

The R script I used to do the analysis is explained here and can also be found on Github.

So, let's get started:

Data Import

The DWD offers a wide range of open-access historical weather data which can be downloaded from their Open Data servers. There are also R packages available to automate this job (e.g. the `rdwd` package), but since I knew exactly which data I needed (working with DWD data is part of my current job), I just went ahead and downloaded the relevant files directly. If you want to read more on how to find your way on the DWD Open Data server, then check out their excellent documentation.

Since I currently live in Leipzig, I downloaded the historic and recent temperature data for the closest DWD station in Leipzig-Holzhausen (station ID 2829) and, for comparison, also the historic and recent temperature data from the oldest continuously operating DWD station in Potsdam (station ID 3987), which is only about 150 km away and should have a reasonably similar climate. If you want to download the data sets yourself, just follow the links that I added here and search for the station ID or use the `rdwd` package:

- historic temperature data
- recent, i.e. this year's, temperature data

I included the current temperature data here as well, as this year's Ice Saints have already passed (and fear not, our seedlings made it safely outside). But that is not necessary, if you only want to use the historical data, i.e. data from 2020 and earlier.

Importing these into R is rather straight forward. After unzipping the files, loading the *tidyverse* bundle of packages, we use the `read_delim()` function to import the data files, as the date is not comma, but semicolon separated. This is very common in German data files as often the comma is used here as decimal separator instead of the decimal point. We then select the columns containing the timestamp and temperature data (again, for a full documentation of the data files, please refer to the DWD documentation), replace error codes for missing values and add a column giving us the location as plain text, i.e. either Leipzig or Potsdam. Lastly, we bind all the tibbles into one, making it easier for us to analyze, group, and nest later on. The *stringr* package will provide us with an easy way to trim white spaces from the data, and we'll use *lubridate* and its new cousin *clock* to help us parse the datetime information throughout the script. For the data files containing the recent temperature data, we also have to aggregate the data from 10 minute intervals to an hourly intervals, as used in the historic data.

```
# Libraries
```

```
#===================================

library(tidyverse)
library(lubridate)
library(clock)
library(stringr)

# Data import
#===================================

temperature_potsdam_hist <-
    # load data
    read_delim(file = "data/produkt_tu_stunde_
18930101_20201231_03987.txt",
               delim = ";") %>%
    rename(temperature = TT_TU) %>%
    mutate(datetime = lubridate::ymd_h(MESS_DATUM),
           temperature = as.numeric(stringr::str_trim(temperature)),
           temperature = ifelse(temperature == -999, NA, temperature))
%>%
    select(datetime, temperature) %>%
    mutate(location = "Potsdam")

temperature_potsdam_recent <-
    # load data
    read_delim(file = "data/produkt_zehn_min_tu_
20191117_20210519_03987.txt",
               delim = ";") %>%
    # extract the temperature column, add NA's, and format the datetime
column
    rename(temperature = TM5_10) %>%
    mutate(datetime = lubridate::ymd_hm(MESS_DATUM),
           temperature = as.numeric(stringr::str_trim(temperature)),
           temperature = ifelse(temperature == -999, NA, temperature))
%>%
    select(datetime, temperature) %>%
    # filter for the current year's data only
    filter(datetime > "2021-01-01 00:00:00") %>%
    # group into hourly averages
    group_by(datetime = lubridate::ceiling_date(datetime, unit =
"hour")) %>%
    summarise(temperature = round(mean(temperature, na.rm = TRUE), 1))
%>%
    # add location label
    mutate(location = "Potsdam")

temperature_leipzig_hist <-
    read_delim(file = "data/produkt_tu_stunde_
19510101_20201231_02928.txt",
               delim = ";") %>%
    rename(temperature = TT_TU) %>%
    mutate(datetime = lubridate::ymd_h(MESS_DATUM),
           temperature = as.numeric(stringr::str_trim(temperature)),
           temperature = ifelse(temperature == -999, NA, temperature))
%>%
    select(datetime, temperature) %>%
    mutate(location = "Leipzig")

temperature_leipzig_recent <-
    # load data
    read_delim(file = "data/produkt_zehn_min_tu_
20191117_20210519_02928.txt",
              delim = ";") %>%
    # extract the temperature column, add NA's, and format the datetime
column
    rename(temperature = TM5_10) %>%
    mutate(datetime = lubridate::ymd_hm(MESS_DATUM),
           temperature = as.numeric(stringr::str_trim(temperature)),
           temperature = ifelse(temperature == -999, NA, temperature))
%>%
    select(datetime, temperature) %>%
    # filter for the current year's data only
    filter(datetime > "2021-01-01 00:00:00") %>%
    # group into hourly averages
    group_by(datetime = lubridate::ceiling_date(datetime, unit =
"hour")) %>%
    summarise(temperature = round(mean(temperature, na.rm = TRUE), 1))
%>%
    # add location label
    mutate(location = "Leipzig")

temperature_hourly =
    bind_rows(
        temperature_potsdam_hist,
        temperature_leipzig_hist,
        temperature_potsdam_recent,
        temperature_leipzig_recent
    ) %>%
        mutate(location = factor(location))
```

This results in a data frame with data starting in 1893 up until basically today. Interesting is also to see the record low and high temperatures for these locations – a chilly minus 26 and a whopping positive 38 degree Celsius!

```
summary(temperature_hourly)
```

```
    datetime                       temperature         location
 Min.   :1893-01-01 01:00:00   Min.   :-26.800   Leipzig: 399112
 1st Qu.:1936-06-25 07:30:00   1st Qu.:  2.800   Potsdam:1125347
 Median :1965-06-24 13:00:00   Median :  8.900
 Mean   :1964-12-26 23:23:40   Mean   :  9.034
 3rd Qu.:1999-08-10 03:00:00   3rd Qu.: 15.000
 Max.   :2021-05-20 00:00:00   Max.   : 38.600
                               NA's   :188
```

Finding the last frost day of the year

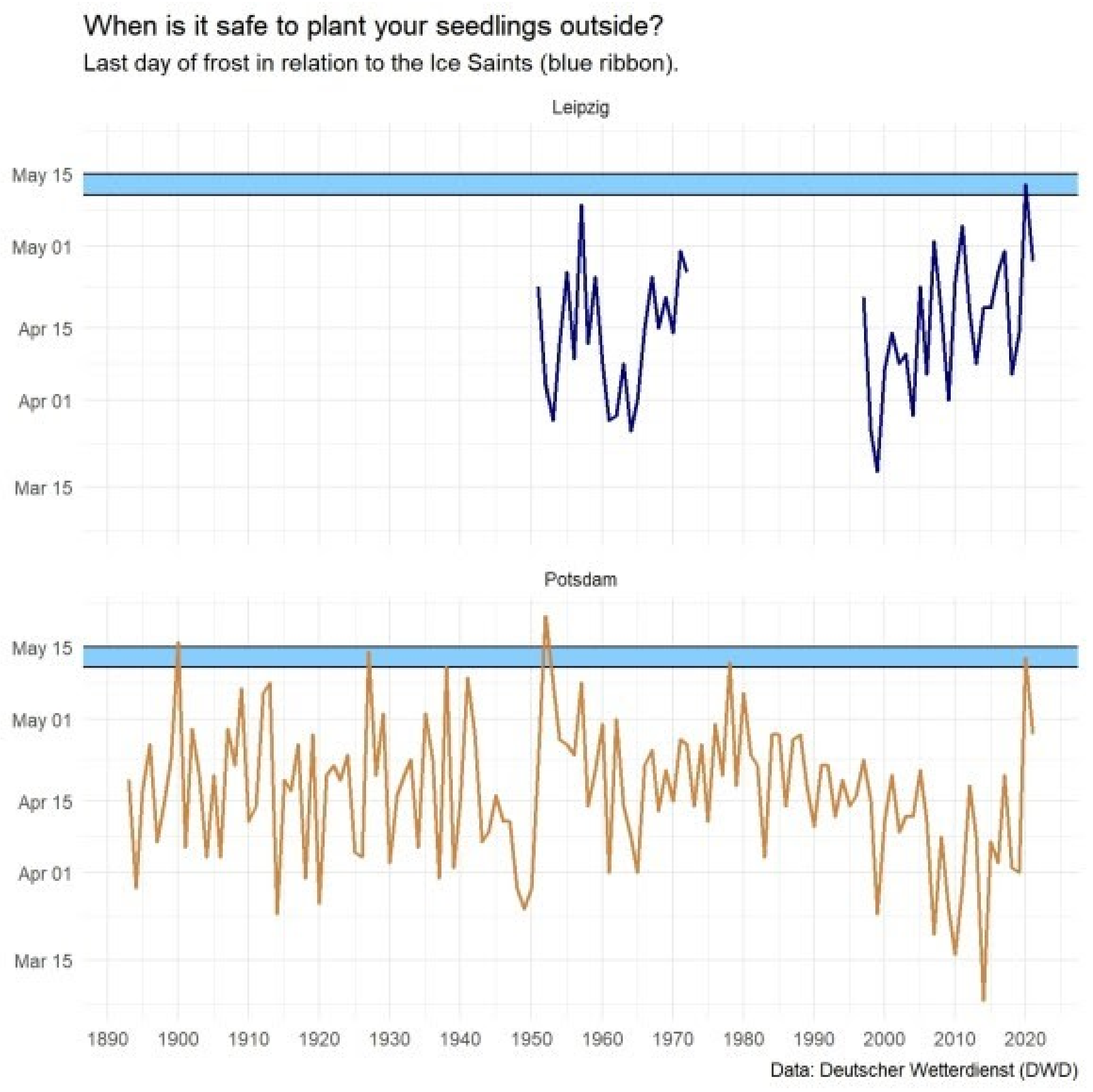
In order to be able to tell when the last day of frost occurred for a given year and location, we first have to tease out the daily minimum temperature and then find the latest day of the year where a subzero temperature ocurred. We have to be careful however, as temperatures in months late in the year easily can reach freezing, so we limit our data set to only days in roughly the first half of the year, i.e. the first 180 days of the year.

```
# Calculate daily minimum temperature
#===================================

temperature_daily_min =
    temperature_hourly %>%
    # calculate deaily min temperature for each location
    drop_na() %>%
    group_by(location,
             date = as_date(datetime)) %>%
    summarise(temp_min = min(temperature)) %>%
    ungroup() %>%
    # add marker if daily min temperaure is below zero
    mutate(subzero = ifelse(temp_min < 0, TRUE, FALSE)) %>%
    mutate(year = lubridate::year(date),
           day_of_year = lubridate::yday(date))


# Calculate last frost day for each year and location
#===================================

last_frost =
    temperature_daily_min %>%
    filter(day_of_year <= 180,
           subzero) %>%
    group_by(location, year) %>%
    summarise(date = max(date),
              day_of_year = lubridate::yday(date)) %>%
    ungroup() %>%
    mutate(day_month = as.Date(day_of_year, origin = "0000-01-01")) %>%
    complete(location, year)
```

Here, I also introduced a sort of dummy date variable `day_month` which only contains information on the month and day, but sets the year equal to zero. This is for plotting and grouping later, as we won't be interested in the specific year, but only the day of the year when the last frost appeared. Of course, you can also create a day-of-year variable directly, i.e. a number between 1 and 365, to code for this, but when plotting the data, I find it more intuitive to see the date in month-day format. Like with most things in R, there is more than one way to achieve a result.

Once we have that out of the way, it's time to have a first look at the data. So, when was the last day of frost in each year for both locations? And how does that compare to the days of the Ice Saints?

```
# Plot historic data on last frost day for each year and location
#===================================
# define start and end of Ice Saints
ice_saints_start = as.Date(131, origin = "0000-01-01")
ice_saints_end = as.Date(135, origin = "0000-01-01")

ggplot(last_frost) +
    geom_rect(aes(xmin = -Inf,
                  xmax = Inf,
                  ymin = ice_saints_start,
                  ymax = ice_saints_end),
              fill = 'skyblue1') +
    geom_hline(aes(yintercept = ice_saints_start)) +
    geom_hline(aes(yintercept = ice_saints_end)) +
    geom_line(aes(x = year,
                  y = day_month,
                  color = location),
              size = 0.8) +
    facet_wrap(~location,
               ncol = 1) +
    scale_x_continuous(breaks = seq(0, 10000, 10)) +
    scale_color_manual(values = c("navy", "peru")) +
    labs(title = "When is it safe to plant your seedlings outside?",
         subtitle = "Last day of frost in relation to the Ice Saints
(blue ribbon).",
         x = NULL,
         y = NULL,
         caption = "Data: Deutscher Wetterdienst (DWD)") +
    theme_minimal() +
    theme(legend.position = "none")
```



When was the last frost day in Leipzig and Potsdam?

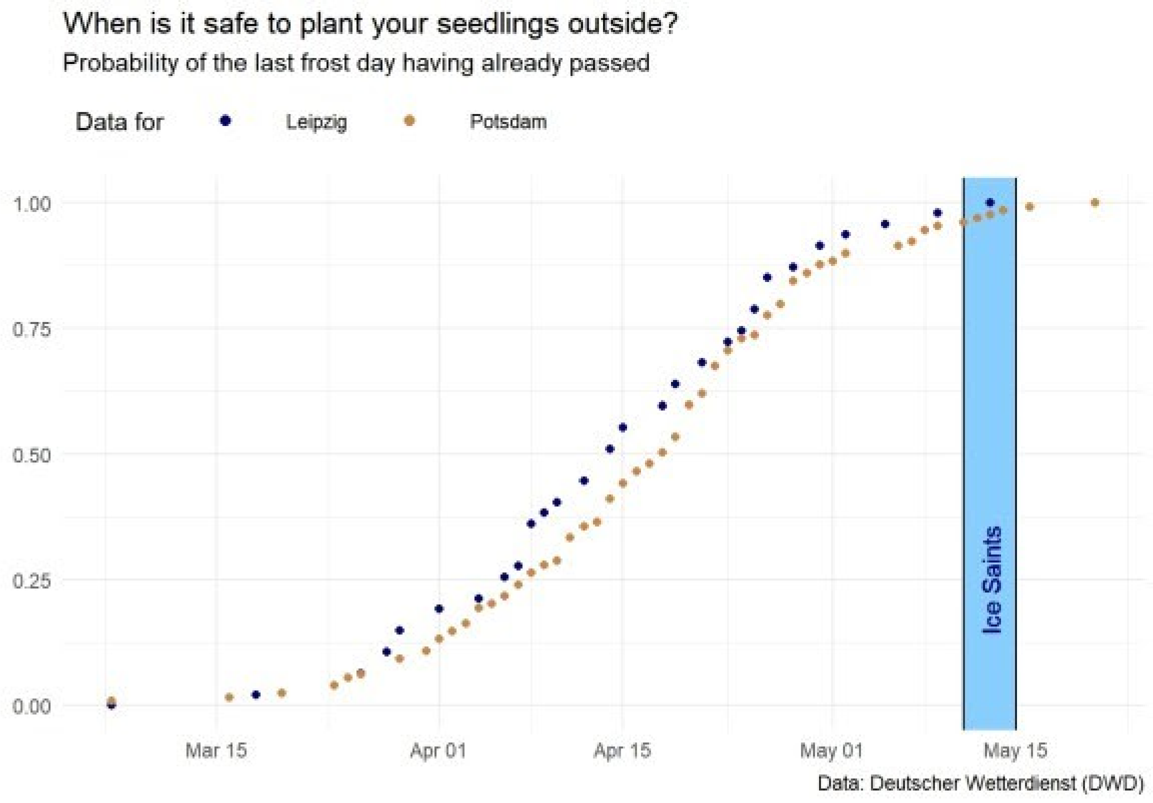
Interestingly, we see that the Ice Saints appear to be a rather good indication of frost free days to follow. Historically, there were only two occasions in Potsdam when a day of frost appeared after the Cold Sophia had passed – in 1900 and in the early 1950's. A few more times it came close, e.g. surprisingly in 2020 for both Leipzig and Potsdam. Despite the fact that there is considerably less data available for Leipzig as compared to Potsdam (if anyone knows what happened between the early 1970's and 1990's with the Leipzig data, do get in touch!), it looks like there was a trend towards earlier frost free days in Potsdam in the last decades: Since 1980 the last frost free day appeared well before May 1st, with a record in 2014, where there was no frost after mid-March. The last two years (2020 and 2021) kind of stopped that trend with cold spells in mid-May and late April, respectively.

Converting to probabilities and fit logit model

Next, we will calculate the probability of frost free nights following each day based on this historic data and feed the data into a logit model to get an estimation when it will be safe to plant the seedlings outside, given a certain risk tolerance. After all, the Ice Saints rule should give you, as a gardener or farmer, complete peace of mind. But if you're ok with a, say, once in a lifetime risk to plant your seedlings out before the last frost, then we'll have to dig a little deeper.

```
# Calculate probability of last frost having passed
```

```
#===================================

last_frost_probability <-
    last_frost %>%
    # remove years without data
    drop_na() %>%
    # calculate the cumulative probability of last frost having passed
    group_by(location, day_month) %>%
    summarise(prob_abs = n()) %>%
    mutate(prob_rel = prob_abs / sum(prob_abs),
           prob_cum = cumsum(prob_rel)) %>%
    ungroup() %>%
    # bring date ranges to a common start and end point
    complete(location, day_month) %>%
    # add explicit missing values and set start end end of the curve
(prob = 0 or 1)
    group_by(location) %>%
    padr::pad(interval = "day") %>%
    mutate(prob_cum = ifelse(day_month == min(day_month) &
is.na(prob_cum), 0, prob_cum),
           prob_cum = ifelse(day_month == max(day_month) &
is.na(prob_cum), 1, prob_cum)) %>%
    ungroup() %>%
    # add a day-of-year-count
    mutate(day_of_year = lubridate::yday(day_month))

# Plot probabilities
#===================================

ggplot(last_frost_probability) +
    geom_rect(aes(xmin = ice_saints_start,
                  xmax = ice_saints_end,
                  ymin = -Inf,
                  ymax = Inf),
              fill = 'skyblue1') +
    geom_vline(aes(xintercept = ice_saints_start)) +
    geom_vline(aes(xintercept = ice_saints_end)) +
    geom_text(aes(x = ice_saints_start + 2,
                  y = 0.25),
              angle = 90,
              color = "darkblue",
              label = "Ice Saints") +
    geom_point(aes(x = day_month,
                   y = prob_cum,
                   color = location),
               size = 1.5) +
    scale_color_manual(values = c("navy", "peru")) +
    labs(title = "When is it safe to plant your seedlings outside?",
         subtitle = "Probability of the last frost day having already
passed",
         x = NULL,
         y = NULL,
         color = "Data for",
         caption = "Data: Deutscher Wetterdienst (DWD)") +
    theme_minimal() +
    theme(legend.position = "top",
          legend.justification ='left',
          legend.key.width = unit(1.5,"cm")) +
    guides(color = guide_legend(override.aes = list(size = 2)))
```



Probability of last frost having passed.

To fit the logit model to the data, we first have to define the model using the `glm()` function. Then, we create a date range for the model predictions/interpolations. Note, that we are only interested in describing the existing data here and we won't be making any predictions for the future. Lastly, we fit the model using nested data frames by location. This nesting of data frames is one of my favorite little tools in R. We then use the defined prediction time range (basically all days from January until the end of May) to calculate the probability of the last frost day having passed for each day using our model, effectively interpolating the available data. We won't be looking into model performance at all here.

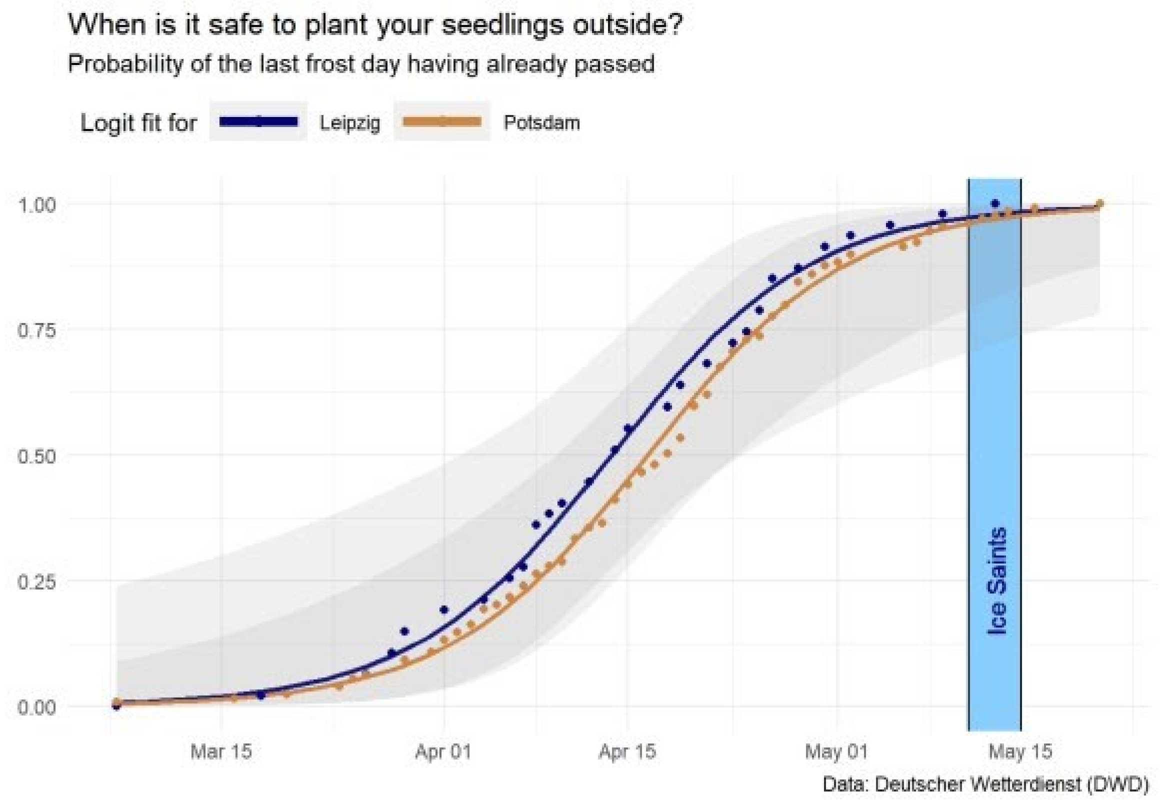
Also, if someone knows a more elegant way to extract the model predictions while keeping the original data in the same data frame directly, do let me know. In the meantime, we do it with a `left_join()`.

```
# Fit the logit model
#===================================

# define logit function
logit_model  <- function(df) {
    glm(prob_cum ~ day_month,
        data = df,
        family = binomial(logit))
}
```

```
# create a date range for the model prediction/interpolation
fit_dates <-
    tibble(day_month =
               date_seq(from = as.Date("0000-01-01"),
                        to = as.Date("0000-05-31"),
                        by = 1))

# fit the model and interpolate
last_frost_model <-
    last_frost_probability %>%
    # subset relevant columns
    select(location, prob_cum, day_month) %>%
    # nest data by location
    group_by(location) %>%
    nest() %>%
    # run logit model and predict on the entire date range
    mutate(model = map(data, logit_model)) %>%
    mutate(fit = map(model, predict, type = "response", newdata =
fit_dates)) %>%
    unnest(fit) %>%
    select(location, fit) %>%
    # add prediction date range
    mutate(day_month = fit_dates$day_month) %>%
    # add original prob_cum column
    left_join(last_frost_probability %>%
                  select(location, day_month, prob_cum),
              by = c("location", "day_month"))
```

And now we can check, how well the model fits our data. Note, that I use a `geom_smoth()` to add the fit based on the same `glm()` function as used in the model calculations, instead of plotting the model results directly. Same, same.

```
# Plot probabilities and model
#===================================

ggplot(last_frost_probability) +
    geom_rect(aes(xmin = ice_saints_start,
                  xmax = ice_saints_end,
                  ymin = -Inf,
                  ymax = Inf),
              fill = 'skyblue1') +
    geom_vline(aes(xintercept = ice_saints_start)) +
    geom_vline(aes(xintercept = ice_saints_end)) +
    geom_text(aes(x = ice_saints_start + 2,
                  y = 0.25),
              angle = 90,
              color = "darkblue",
              label = "Ice Saints") +
    geom_smooth(aes(x = day_month,
                    y = prob_cum,
                    color = location),
                alpha = 0.15,
                method = "glm",
                method.args = list(family = binomial(logit))) +
    geom_point(aes(x = day_month,
                   y = prob_cum,
                   color = location),
               size = 1.5) +
    scale_color_manual(values = c("navy", "peru")) +
    labs(title = "When is it safe to plant your seedlings outside?",
         subtitle = "Probability of the last frost day having already
passed",
         x = NULL,
         y = NULL,
         color = "Logit fit for",
         caption = "Data: Deutscher Wetterdienst (DWD)") +
    theme_minimal() +
    theme(legend.position = "top",
          legend.justification ='left',
          legend.key.width = unit(1.5,"cm")) +
    guides(color = guide_legend(override.aes = list(size = 2)))
```



Probability of last frost having passed with fit.

With this data, we can now compile our risk table, i.e. an overview of when you can plant out your seedlings, given a certain risk you are willing to tolerate based on the historic data. This is also a great excuse for me, to finally have a go at the *gt* package for making nice tables for documents in R. I also use the `divide_by()` from the *magrittr* package which is most famous for providing the pipe operator `%>%`, but also has plenty of pipe friendly operators.

```
library(magrittr)
library(gt)
```

```
# Create summary table based on model results
#==================================

prob_table =
    last_frost_model %>%
    group_by(location) %>%
    mutate(above_50 = fit >= 0.5,
           above_90 = fit >= 0.9,
           above_95 = fit >= 0.95,
           above_98 = fit >= 0.98,
           above_99 = fit >= 0.99) %>%
    ungroup() %>%
    select(location, day_month, starts_with("above")) %>%
    gather(key = prob,
           value = response,
           starts_with("above")) %>%
    filter(response) %>%
    group_by(location, prob) %>%
    summarise(threshold = min(day_month)) %>%
    ungroup() %>%
    mutate(prob = str_remove(prob, "above_") %>%
               as.numeric() %>%
               magrittr::divide_by(100)) %>%
    spread(key = location,
           value = threshold) %>%
    mutate(explainer = paste0("i.e., you can expect a frost day after
this date once in ", round(1/(1-prob), 0), " years"))

gt(prob_table) %>%
    tab_header(title =  md("**When is it safe to plant your seedlings
outside?**"),
               subtitle = "Probability of the last frost day having
already passed") %>%
    opt_align_table_header(align = "left") %>%
    tab_spanner(label = "City",
                columns = vars("Leipzig", "Potsdam")) %>%
    cols_label(prob = "Probability",
               explainer = "") %>%
    fmt_percent(columns = vars(prob),
                decimals = 0) %>%
    fmt_date(columns = vars("Leipzig", "Potsdam"),
             date_style = 9) %>%
    cols_align(align = "center",
               columns = everything())
```

When is it safe to plant your seedlings outside?

Probability of the last frost day having already passed

Probability	**City**		
	Leipzig	**Potsdam**	
50&percnt;	14 April	17 April	i.e., you can expect a frost day after this date once in 2 years

When is it safe to plant your seedlings outside?

Probability of the last frost day having already passed

90&percnt;	1 May	4 May	i.e., you can expect a frost day after this date once in 10 years
95&percnt;	7 May	9 May	i.e., you can expect a frost day after this date once in 20 years
98&percnt;	14 May	17 May	i.e., you can expect a frost day after this date once in 50 years
99&percnt;	19 May	22 May	i.e., you can expect a frost day after this date once in 100 years

While you can seemingly plant out your seedlings two to three days earlier in Leipzig than Potsdam independent of your risk tolerance, this table can now guide you (or rather me). If we assume a once-a-lifetime risk of me planting the seedlings out to early, then I should not plant out before May 14 (at 2% probability that there still could be a frost day, i.e. once in 50 years). If you’re very risk tolerant and your seedling are very frost tolerant, then you could already chance it and plant them out in mid-April for a 50/50 chance of frost after that date.