Exercise sheet 4
Prepare the below such that you are able to discuss it on Wednesday 24th March

Exercise 1 Back calculation

The nonparametric back-projection method is implemented in the function `backprojNP` in the R package **surveillance**.

a) Load the R package **surveillance** and read the documentation of the `backprojNP` function

b) Run `example(backprojNP)` and ensure you understand the inputs and outputs at each step. Feel free to use the OLAT forum to discuss with each other.

Based on a study of haemophiliacs, Brookmeyer and Goedert (1989) suggest the incubation time from exposure to HIV to onset of AIDS can be described by a Weibull distribution with survival function

$$S(t) = \exp\left(-0.0021 \left(\frac{t}{4}\right)^{2.516}\right), \quad t \geq 0,$$

Using this expression, the probability mass function (PMF) of the incubation time distribution D obtained by discretising the above and right-truncating it after 30, gives the `pmf` object:

```
#Weibull in Brookmeyer und Goedert (1989)
F <- function(t) {
  ifelse(t >= 0, 1 - exp( - 0.0021 * t ** 2.516), 0)
}
t_grid <- seq(- 1, 30 * 4, by = 1)
pmf <- F(tail(t_grid / 4, n = - 1)) - F(head(t_grid / 4, n = - 1))
pmf <- pmf / sum(pmf)
```

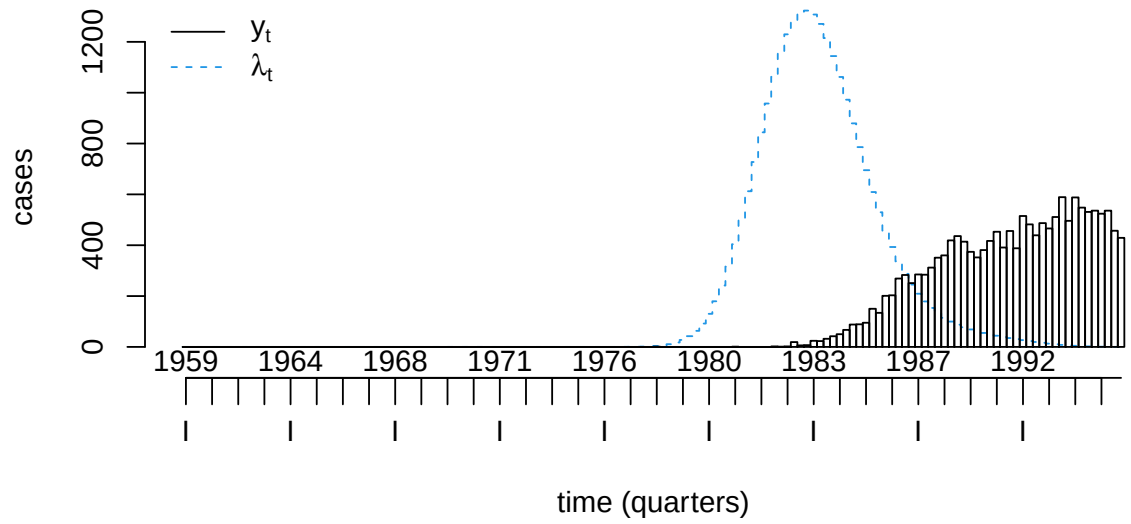
a) Run `backprojNP` with this PMF to perform a back projection of the AIDS incidence with a smoothing of $k = 4$ with the control option `eq3a.method = "C"`
Solution: We use the AIDS data provided and include some smoothing since HIV/AIDS is not a point-source outbreak (as the *E.coli* example in the lecture with `k = 0`) but infectious

```
library(surveillance)
bp <- backprojNP(aids.sts, incu.pmf = pmf,
                 control = list(k = 4,
                                verbose = FALSE,
                                eps = 0.00001,
                                iter.max = 1000,
                                eq3a.method = "C"))
```

b) Plot the estimated λ_t's as a function of time and interpret the result.
Solution:

```
plot(bp, xlab = "time (quarters)", ylab = "cases",
     legend.opts = NULL, main = "",
```

```
     xaxis.tickFreq = list("%Y" = atChange))
legend(x = "topleft", c(expression(y[t]), expression(lambda[t])),
       lty = c(1, 2), col = c(1, 4), bty = "n")
```



Recall the goal of back-proejction is to infer the time of infection from time of symptom onset and use this information to reconstruct the infection curve; deduce λ_t given observed cases y_t. We see that the peak of λ_t is earlier than the peak of y_t (as expected) but also that λ_t seems to be narrower than y_t. NB additional context: from 1996 the use of antiretroviral therapy changed the incubation time which is why we are only analysing data until 2015.

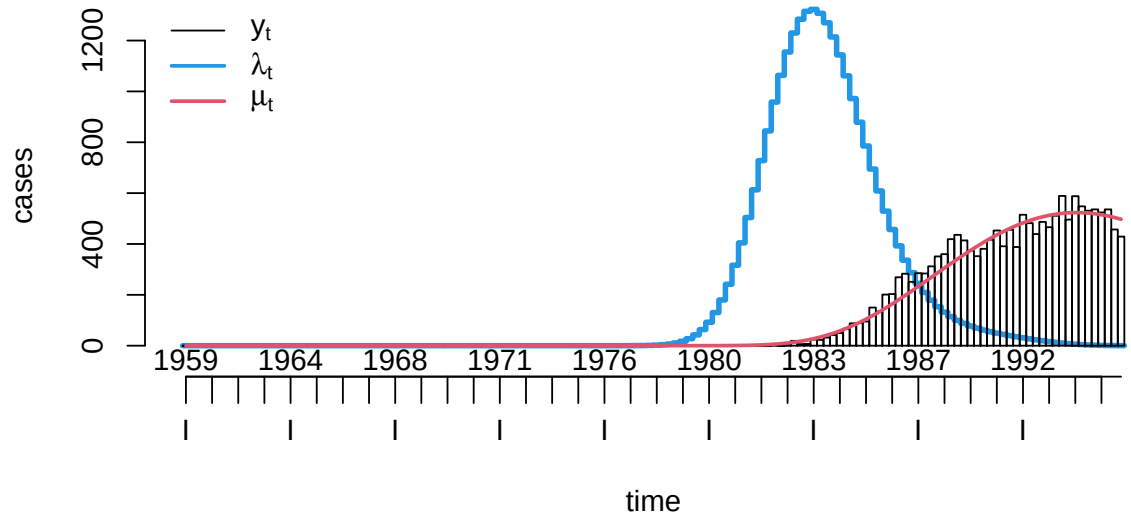
c) Use the estimated λ_t's to obtain an estimate of μ_t for $t = 1, \ldots, 144$. Create a plot containing λ_t, the observed number of AIDS patients y_t (available via `aids$observed`, and μ_t as a function of time t.
Solution: Recall from the lecture that μ_t is the mean in $Y_t \sim \text{Po}(\mu_t)$. We calculate

$$\mu_t = \sum_{i=1}^{t} f(t-i)\lambda_i$$

where f is the PMF

```
#Create wrapper functions for the PMF and CDF based on the vector
#This safeguards queries outside the support of the pmf.
dincu <- function(x) {
  notInSupport <- x < 0 | x >= length(pmf)
  #Give index -1 to invalid queries
  x[notInSupport] <- -1
  return(c(0, pmf)[x + 2])
}

#Extra estimated lambdas
lambda <- as.numeric(upperbound(aids.sts))
```

```r
#Do convolution to obtain mu
mu <- 0 * lambda
for (t in 1 : length(mu)) {
  delay <- 0 : (t - 1)
  mu[t] <- sum(lambda[t - delay] * dincu(delay))
}

plot(aids.sts, legend.opts = NULL, lwd = c(1, 1, 3),
     lty = c(1, 1, 1), ylab = "cases",
     dx.upperbound = 0, main = "",
     xaxis.tickFreq = list("%Y" = atChange))
lines(1 : length(mu), mu, col = 2, lwd = 2)
legend(x = "topleft", c(expression(y[t]), expression(lambda[t]),
                        expression(mu[t])),
       lty = c(1, 1, 1), col = c(1, 4, 2),
       lwd = c(1, 2, 2), bty = "n")
```



It fits well to the observed data

d) What assumptions are required to compute future values μ_t for $t = 145, 146, 147, 148$?
Solution: We need to assume that

- the incubation distribution is the same as before
- $\lambda_{145} = \ldots = \lambda_{148} = 0$ i.e. after 1995:IV no new cases occur

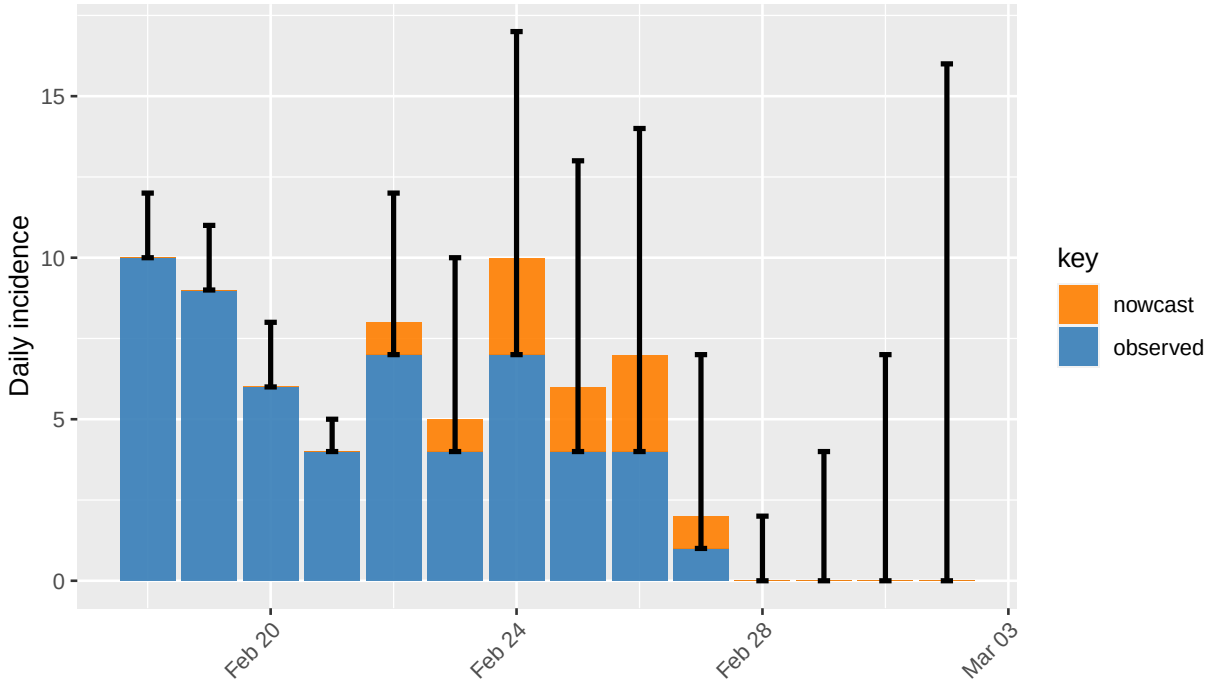
Neither of these assumptions are realistic given what we know (see earlier contextual comment)

Exercise 2 Nowcasting

a) With the data available here, use the script provided on OLAT (`script.R`) to examine what

happens when the moving window width is changed? Try both increasing and decreasing it. What happens when you remove that option from the nowcast controls?
Hint: Reading the help file for **surveillance**::`nowcast` may help
Solution: The default output is

```
## Building reporting triangle...
## No. cases:  131
## No. cases within moving window:  62
## bayes prep...
## (E,V) of prior for lambda = ( 2.18333333333333,31.2659478438957 )
## bayes.trunc...
```

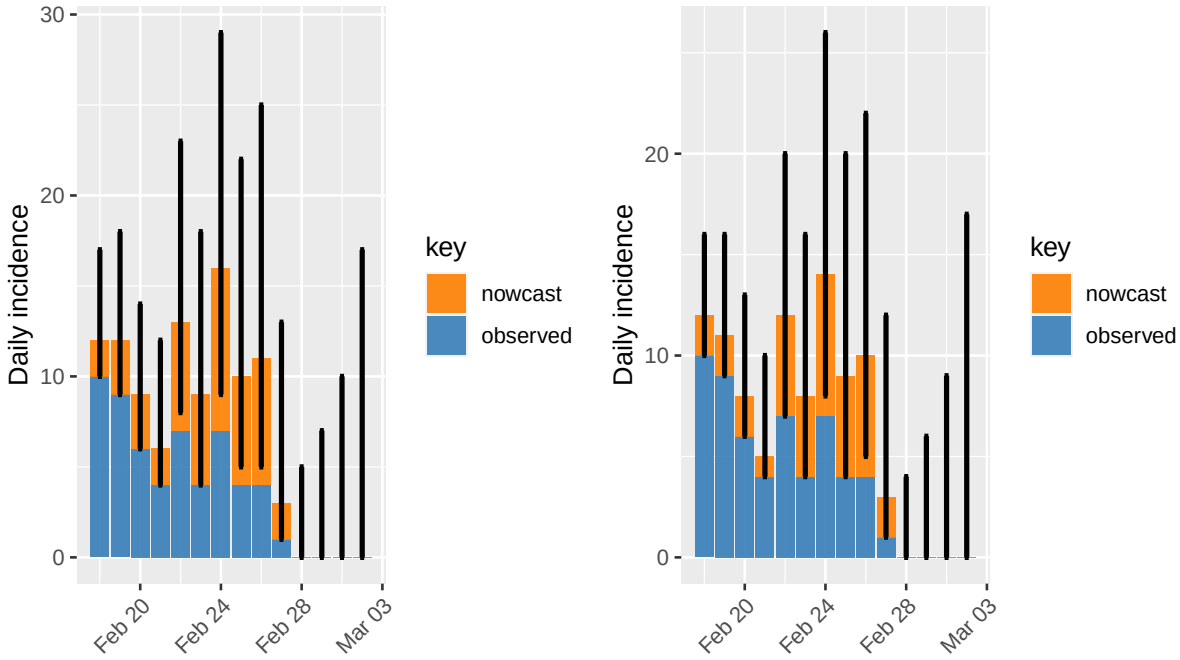


We double the moving window as well as remove it. These options (`m = 28` and `m = NULL`, respectively) to the largest possible option in the `nowcast` call yields no change in the plots but different numbers of cases within the windows are seen in the output

```
nc <- nowcast(now = now, when = nowcastDates, data = as.data.frame(df),
              dEventCol = "date_onset_symptoms",
              dReportCol = "date_confirmation",
              aggregate.by = "1 day",
              D = 14, # adjust cases up to 2 weeks back.
              # # Assume constant delay distribution, but only within the last m=1
              method = "bayes.trunc",
              m = NULL,
              control = nc.control
)

## Building reporting triangle...
## No. cases:  131
## No. cases within moving window:  131
## bayes prep...
## (E,V) of prior for lambda = ( 2.18333333333333,31.2659478438957 )
## bayes.trunc...
```

```
## Building reporting triangle...
## No. cases:  131
## No. cases within moving window:  103
## bayes prep...
## (E,V) of prior for lambda = ( 2.18333333333333,31.2659478438957 )
## bayes.trunc...
```

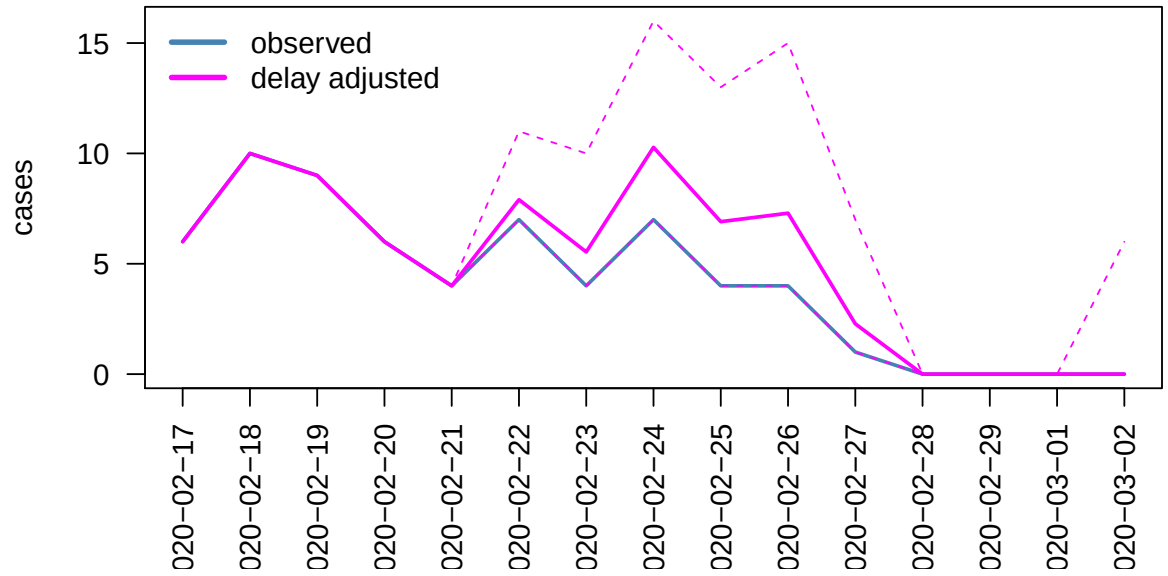
```r
# Save plot as object
p2 <- nc_tidy %>%
  mutate(key = case_when(key == "obnyr" ~ "nowcast",
                         TRUE ~ key)) %>%
  filter(date <= ymd(last_linelist_case)) %>%

  ggplot(aes(date, value)) +
  geom_col(aes(fill = key), alpha = 0.9)  +
  geom_errorbar(
    data = (nc_df %>%
              mutate(value = 1, key = NA) %>%
              filter(date > (max(date) - weeks(2)))),
    aes(ymin = predicted_lower, ymax = predicted_upper),
    width = 0.2, size = 1) +
  scale_fill_manual(values = c("#ff7f00", "#377eb8")) +
  scale_y_continuous(labels = scales::number_format(accuracy = 1)) +
  scale_x_date(date_breaks = "4 days", date_labels = "%b %d") +
  labs(x = "",
       y = "Daily incidence") +
  theme(axis.title.x = element_blank(),
        axis.text.x = element_text(angle = 45, hjust = 1))

library(patchwork)
p1 + p2
```



b) Use the `glm` function to generate a nowcast for the time points between for the Japan data from the previous question and plot your nowcast
Hint: follow the code from the lecture slides
Solution:

```
# From the help function - extract the reporting triangle
nc <- nowcast(now = now, when = nowcastDates, data = as.data.frame(df),
              dEventCol = "date_onset_symptoms",
              dReportCol = "date_confirmation",
              aggregate.by = "1 day",
              D = 14, # adjust cases up to 2 weeks back.
              # # Assume constant delay distribution, but only within the last m=1
              method = "unif")

## Building reporting triangle...
## No. cases:  131
## No. cases within moving window:  131

zeger <- nc@reportingTriangle
idx <- seq(which(attr(zeger, "t02s") == min(nowcastDates)), nrow(zeger))
zeger <- zeger[idx, ]

matrix2df <- function(zeger){
  data.frame(n = as.numeric(as.matrix(zeger)),
             t = as.numeric(as.matrix(row(zeger) - 1)),
             d = as.numeric(as.matrix(col(zeger) - 1)))
}

#Convert to data.frame
zeger_df <- matrix2df(zeger)
#Fit log-linear model.
m <- glm(n ~ as.factor(t) + as.factor(d),
         data = zeger_df, subset = !is.na(n),
         family = poisson)

#Prediction m_{t,d} for ALL cells in the contingency table
mu_mle <- predict(m, newdata = zeger_df, type = "response")

NtInf <- function(data){
    as.numeric(with(data, tapply(n, t, sum, na.rm = TRUE)))
}

#Function to generate new data by parametric bootstrap
rntd <- function(data, mle){
    #Indicator vector of what is observed
    observed <- !is.na(data$n)
    #Extra data copies (one to estimate, one to predict)
    data_estimate <- data_predict <- data

    #Make a new data matrix with observed values replaced
    data_estimate$n[observed] <- rpois(n = nrow(data),
                                       lambda = mle)[observed]
```

```r
    #Fit Poisson GLM to the data to obtain estimates
    m_star <- glm(n ~ as.factor(t) + as.factor(d),
                  data = data_estimate, subset = !is.na(n),
                  family = poisson)

    #Add sampled values where missing
    data_predict$n[!observed] <- rpois(n = nrow(data),
                                       predict(m_star, newdata = data,
                                               type = "response"))[!observed]
    #Done - return new data.frame
    return(data_predict)
}

set.seed(20210324)
b <- boot::boot(zeger_df, statistic = NtInf,
                sim = "parametric", R = 999,
                ran.gen = rntd, mle = mu_mle)

#Simple percentile intervals
predIntervals <- apply(rbind(b$t0, b$t), 2,
                       quantile, prob = c(0.025, 0.975))


## ------------------------------------------------------------------------
#Perform nowcasting only based on the mean
zeger_df2 <- zeger_df
zeger_df2$n[is.na(zeger_df2$n)] <- mu_mle[is.na(zeger_df2$n)]

plot(1 : nrow(zeger), b$t0, type = "l",
     ylab = "cases", ylim = c(0, max(predIntervals)),
     axes = FALSE, xlab = "", lwd = 2, col = "steelblue")
axis(2, las = 1)
axis(1, at = 1 : nrow(zeger),
     label = rownames(zeger), las = 2)
box()
lines(1 : nrow(zeger), NtInf(zeger_df2),
      col = "magenta", lwd = 2)
matlines(1 : nrow(zeger), t(predIntervals),
         lty = 2, col = "magenta", lwd = 1)
legend(x = "topleft", c("observed", "delay adjusted"),
       lty = c(1, 1), col = c("steelblue", "magenta"),
       lwd = 3, bty = "n")
```



We see a similar pattern to that found in the nowcast from the first exercise.