Two exercises in computational statistics

[redacted]

1 Exercise 1

1.1 Introduction

We fit a generalized linear model (GLM) to predict the number of visits to a family doctor for a dataset of 1204 individuals. The model uses an intercept, plus a linear combination of the individual's age (`age`), net monthly household income measured in thousands of German Marks (`hhninc`), whether the individual is female (`female`), whether the individual is living in a household with kids under the age of sixteen (`hhkids`), how many years of schooling the individual has (`educyrs`), and whether the individual has additional health insurance (`addins`). The three binary variables `female`, `hhkids`, and `addins` are given as 1 or 0, where 1 indicates affirmatively that the individual is female, lives in a household with kids, or has additional health insurance, respectively.

The number of doctor visits is a count of events, presumably within a fixed time interval. Therefore, the Poisson distribution is a reasonable choice for the generalized linear model. A natural way to link the linear combination of predictor variables (η_i) and the mean rate of occurrence for each observation (μ_i) is the canonical link function. Thus, our model assumes $Y_i \sim \text{Poisson}(\mu_i)$, with $\mu_i = \exp(\eta_i)$.

1.2 Analysis

We obtain the maximum likelihood estimates for the coefficients of the linear predictors and intercept. We also obtain the estimated standard errors of the coefficients of the GLM, which we refer to as the general standard errors (SEs). The general standard errors are the diagonal of the covariance matrix of the coefficients, which for the Poisson model is given by $(X^T \hat{W} X)^{-1}$ where $\hat{W}$ is the diagonal matrix $\text{diag}(\hat{\mu}_1, \hat{\mu}_2, \ldots, \hat{\mu}_n)$. The standard errors are important because under the asymptotic normality of the coefficient estimate they determine the significance of the coefficients through the Wald test. To allow for the possibility of heteroskedasticity, we also use the sandwich estimator of the variance of the coefficients estimates, which is robust to heteroskedasticity. We call these estimates the robust SEs.

The general and robust SEs are shown are Table 1. Importantly, we notice that each of the robust standard errors is greater (by about 30 - 60%) than the general SEs. This suggests that when we drop the assumption that the residuals all have the same variance, the uncertainties in the coefficient estimates are greater, indicating that there may be less confidence that the coefficients are predictive (i.e. significantly different than zero under the Wald test).

The standard errors can also be obtained through the bootstrap variance estimates. That is, B bootstrap samples are taken of the original dataset and the population variance of the bootstrap

coefficient estimates, given as

$$\text{se}(\hat{\beta}_k) = \sqrt{\frac{1}{B}\Sigma_{j=1}^{B}(\hat{\beta}_k^{(j)} - \frac{1}{B}\Sigma_{j=1}^{B}\hat{\beta}_k^{(j)})^2}$$

where $\hat{\beta}_k^{(j)}$ is the estimate of the k-th coefficient using the j-th bootstrap sample. The bootstrap SEs are shown for $B = 10{,}000$ in Table 1. They are much closer to the robust SEs than the general SEs, which suggests that it is more appropriate to allow for heteroskedasticity.

Table 1: The general, robust, and bootstrap estimated standard errors of the Poisson GLM.

Variable	General SE	Robust SE	Bootstrap SE
(Intercept)	0.182105	0.256142	0.255912
age	0.002273	0.003337	0.003273
hhninc	0.015997	0.020989	0.020952
female	0.048738	0.069106	0.068472
hhkids	0.053270	0.074689	0.074743
educyrs	0.010740	0.014560	0.014559
addins	0.126796	0.202008	0.204341

We can also obtain the bootstrap estimates of the coefficients, which is the sample mean of the coefficients from each bootstrap sample, given as

$$\hat{\beta}_k = \frac{1}{B}\Sigma_{j=1}^{B}\hat{\beta}_k^{(j)} \, .$$

From these estimates of the coefficients, we can estimate the bias by subtracting the general coefficient estimates (using all the data) from the bootstrap coefficients – shown in Table 2.

Table 2: The general and bootstrap estimated coefficients of the model, and the estimated bootstrap bias.

Variable	General Coefficients	Bootstrap Coefficients	Bootstrap Bias
(Intercept)	0.364480	0.364653	0.000172
age	0.008266	0.008261	-0.000004
hhninc	-0.064221	-0.064908	-0.000687
female	0.337526	0.33813	0.000604
hhkids	-0.113012	-0.113681	-0.000669
educyrs	-0.022910	-0.022998	-0.000088
addins	0.406207	0.385118	-0.021088

The greater bootstrap and robust SEs imply lower wald test statistics for the significance of the coefficients, which has implications for model selection. For the hhkids and educyrs variables, we can obtain the general and robust Wald test statistic to test whether both these coefficients are zero, where the Wald test statistic is given by

$$W = \begin{pmatrix} \hat{\beta}_{hhkids} \\ \hat{\beta}_{educyrs} \end{pmatrix}^T \left(\hat{var} \begin{pmatrix} \hat{\beta}_{hhkids} \\ \hat{\beta}_{educyrs} \end{pmatrix} \right)^{-1} \begin{pmatrix} \hat{\beta}_{hhkids} \\ \hat{\beta}_{educyrs} \end{pmatrix}$$

such that the middle term is the general, robust, or bootstrap estimated covariance matrices for hhkids and educyrs. We also obtain the Wald test statistic for the bootstrap coefficients and covariance matrix, where the estimated covariance between coefficients k and l is given as

$$\frac{1}{B}\Sigma_{j=1}^{B}(\hat{\beta}_k^{(j)} - \hat{\beta}_k)(\hat{\beta}_l^{(j)} - \hat{\beta}_l) \, ,$$

which uses the earlier bootstrap coefficient estimates $\hat{\beta}_k$ and $\hat{\beta}_l$.

Under a Wald test of multiple parameters (two in this case), the test statistic is compared to a χ^2_d distribution, where d is the number of parameters tested ($d = 2$). The Wald test statistic values and the associated p-values are given in Table 3. The robust and bootstrap p-values are nearly identical and larger than the general p-value, which is unsurprising given that the robust and bootstrap standard errors were nearly identical and larger than the general standard errors (Table 1). At the 5% level, the null hypothesis that the coefficients for `hhkids` and `educyrs` are both zero would be rejected under the general SEs, but would fail to be rejected under the robust and bootstrap SEs. Bootstrap sampling makes the crucial approximation of the empirical cdf to the true cdf, such that the bootstrap samples are random iid draws from the empirical cdf. If this approximation holds, then the bootstrap tells us that under the true distribution of the data, the variance of the coefficients is too large to be able to conclude that they are non-zero.

Table 3: Wald test statistics and associated χ^2_2 p-values for the general, robust, and bootstrap covariance estimates, for testing H_0: $\beta_{hhkids} = \beta_{educyrs} = 0$.

	General	**Robust**	**Bootstrap**
Wald Statistic	8.70927	4.78588	4.72573
p-value	0.01285	0.09136	0.09415

A bootstrap p-value can be calculated for the Wald test statistic based on the robust SEs. Bootstrap p-values are discussed in Allen, Burgess, and Windmeijer (2009). In this case, the bootstrap p-value is

$$\frac{1}{B}\Sigma_{j=1}^{B}\mathbb{1}(W_j > W)$$

where $\mathbb{1}(\cdot)$ is the indicator function and W_j is the relevant Wald test statistic of the j-th bootstrap sample, given as

$$W_j = \begin{pmatrix} \hat{\beta}^{(j)}_{hhkids} - \hat{\beta}_{hhkids} \\ \hat{\beta}^{(j)}_{educyrs} - \hat{\beta}_{educyrs} \end{pmatrix}^T \left(\hat{var} \begin{pmatrix} \hat{\beta}_{hhkids} \\ \hat{\beta}_{educyrs} \end{pmatrix} \right)^{-1} \begin{pmatrix} \hat{\beta}^{(j)}_{hhkids} - \hat{\beta}_{hhkids} \\ \hat{\beta}^{(j)}_{educyrs} - \hat{\beta}_{educyrs} \end{pmatrix}$$

where the robust covariance matrix is used as the middle term. The bootstrap p-value is 0.0905. A histogram of the bootstrap wald statistics is shown in Figure 1. The histogram of Wald test statistics approximates the distribution of Wald test statistics assuming iid draws from the true distribution (given by the approximation of the empirical cdf to the true cdf) and assuming the coefficients β_{hhkids} and $\beta_{educyrs}$ are zero. This latter assumption is obtained by subtracting the general coefficient estimates $\hat{\beta}_{hhkids}$ and $\hat{\beta}_{educyrs}$ from each bootstrap coefficient. The p-value is similar to the bootstrap and robust p-values given in Table 3 and the histogram of Wald test statistics follows the χ^2_2 distribution, as expected.

1.3 Conclusions

Allowing for general forms of heteroskedasticity (by using the robust estimates) leads to larger SE estimates than under the assumption of no heteroskedasticity, which is the case for the general estimated SEs. If the data was non-heteroskedastic, we would expect the robust SEs to be approximately equal to the general SEs. Interestingly, the bootstrap SEs highly approximated the robust SEs. Because the bootstrap SEs are an unbiased estimate of the true SEs, our result implies that the assumption of non-heteroskedasticity is erroneous and the robust SEs are the appropriate estimate.

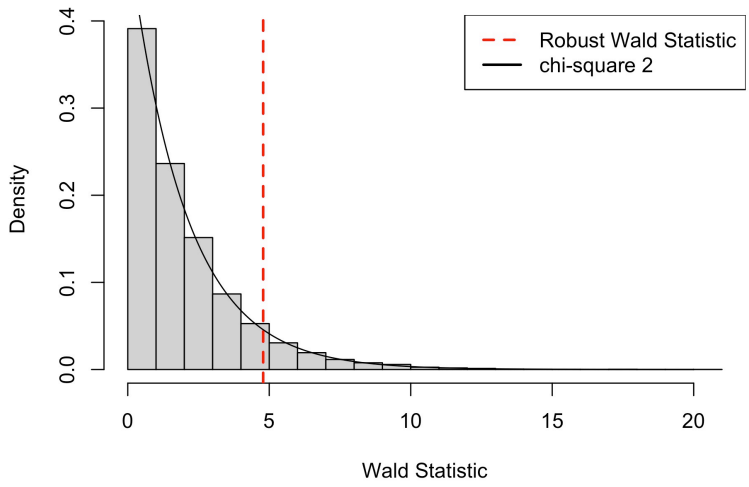


Figure 1: Histogram of the bootstrap wald statistics from the $B = 10{,}000$ bootstrap samples and the robust wald statistic (using all the data) shown with a red dashed line.

It should be noted that under the Poisson distribution, the variance is equal to the mean. Therefore, as the predicted mean increases, so should the variance about it. We should then expect heteroskedasticity in the raw residuals of a Poisson GLM, such that the variance of the residuals increases with (and is equal to) the predicted response. However, the Pearson residuals take into account the variance depending on the mean, thus heteroskedasticity observed in a plot of the Pearson residuals should suggest that there is heteroskedasticity beyond what is expected under the Poisson distribution. Plots of the Pearson and deviance residuals are shown in Figure 2. It can be seen in the plot of the Pearson residuals that the variance of the residuals appears to decrease with the linear predictor. The same trend can be seen, though less clearly, in the plot of the deviance residuals. Therefore, the assumption of non-heteroskedasticity is refuted in the plots of the residuals.

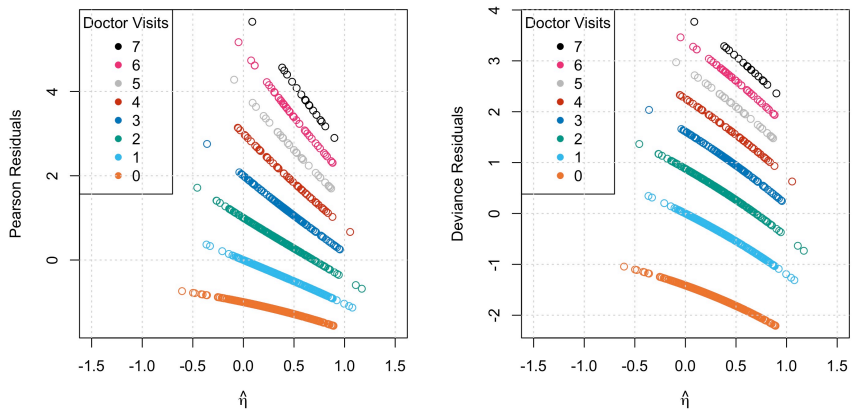


Figure 2: The linear predictor ($\hat{\eta}$) plotted against the Pearson and Deviance residuals, coloured by the number of doctor visits.

Larger SEs directly affect model selection by increasing the p-values for tests of significant (non-zero) coefficients. Accordingly, we saw that the coefficients for `hhkids` and `educyrs` were no longer both significant at the 5% level under the robust and bootstrap variance estimates, and under the bootstrap p-value. These results suggest that it is important to check for heteroskedasticity when undergoing model selection, which can be accomplished by using the robust estimator or bootstrapping.

2 Exercise 2

2.1 Data Exploration

This data consists of the annual inflation rate in France for the years 1956-2020. Exploratory data analysis is given by Figure 3, which shows the inflation rate for each of the available years and the relative value of one denomination in (the start of) 1956. For example, goods and services costing one denomination in 1956 would cost about 14 denominations in 2020. It is apparent in the plot of the inflation rate over time that there is a degree of smoothness, that is, the inflation rates of adjacent years correlate with one another. This suggests the use of an autoregressive (AR) model which uses a linear combination of previous data points to predict the next data point.

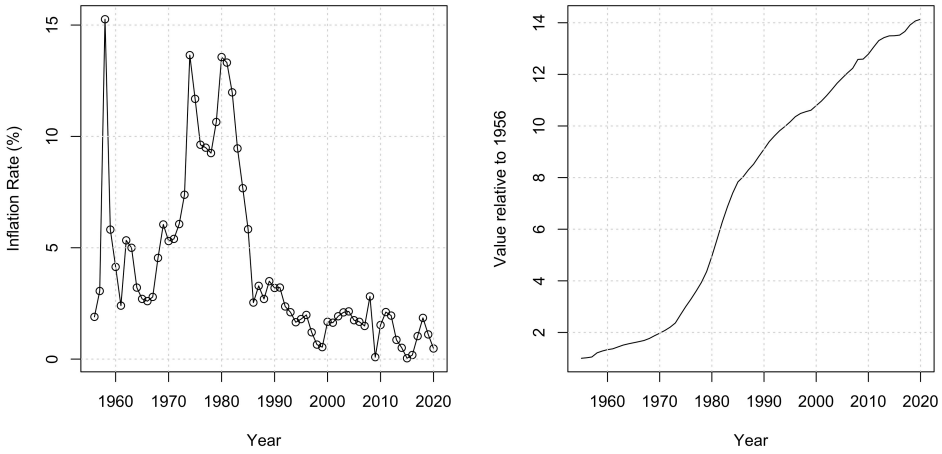


Figure 3: Inflation rate over time and relative value of one denomination in 1956.

2.2 Autoregressive Model

The simplest autoregressive model is an AR(1) model, which uses only the previous (one) data point to predict the next one, given by

$$Y_t = \alpha + Y_{t-1}\beta + W_t$$

where W_t is the random noise of the t-th observation, which has zero mean and constant variance. A natural way to estimate the parameters α and β of the AR(1) model is by minimizing the empirical risk under the squared loss function. This is equivalent to ordinary least squares and maximizing the likelihood of the observations under normally distributed white noise W_t.

A couple of comments can be made on the AR(1) model specification. First, $\alpha = 0$ and $\beta = 1$ is identical to a random walk. Second, changing $\alpha \neq 0$ (whilst keeping $\beta = 1$) produces a random walk with a constant average step of α. It is straightforward to show from basic properties of covariance that the correlation between Y_t and Y_{t-1} is given by $\rho = \frac{\sigma_t}{\sigma_{t-1}}\beta$ where σ_t and σ_{t-1} are the standard deviations of Y_t and Y_{t-1}, respectively. If Y_t and Y_{t-1} have the same marginal distribution, which is reasonable to expect, then they have the same standard deviations and, simply, $\rho = \beta$. That is, the coefficient in the AR(1) model is the correlation between adjacent points. The correlation can be seen for the inflation data by plotting y_t against y_{t-1} as shown in Figure 4. There is clearly a positive correlation suggesting that a linear model is reasonable – although there are many data points in the bottom left corner that are not well fit by the linear model.

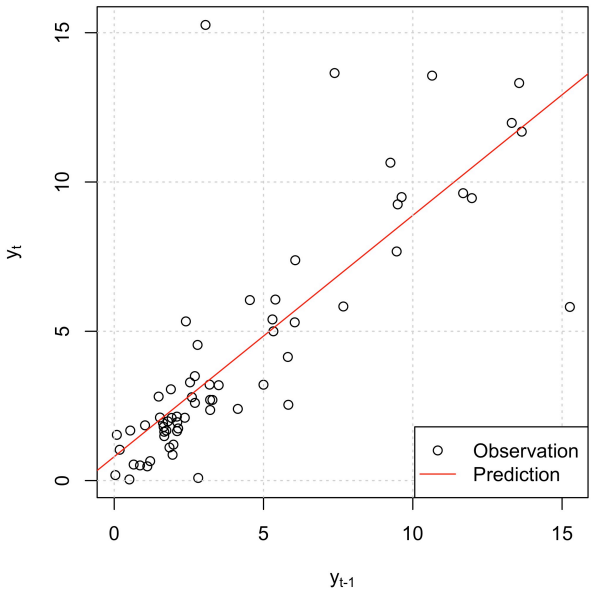


Figure 4: Scatter plot of the inflation rate at year t against the inflation rate at year $t-1$, with the red line demonstrating the predictions of the AR(1) model with non-varying coefficients.

An important feature of the AR(1) model is that the intercept and coefficient are static, such that the correlation between adjacent data points (β), and the average drift (α) cannot vary as a function of time. The inadequacy of the static nature of this model is clear from Figure 5, which shows the observations and predictions of the AR(1) model. The predictions are biased such that they are almost always too high from 1990-2020 (corresponding to the points in the bottom left of Figure 4 as previously mentioned), and the predictions are typically too low for the years 1970-1985. The non-varying coefficients appear to be too simple of a hypothesis class to adequately model the inflation rate over time.

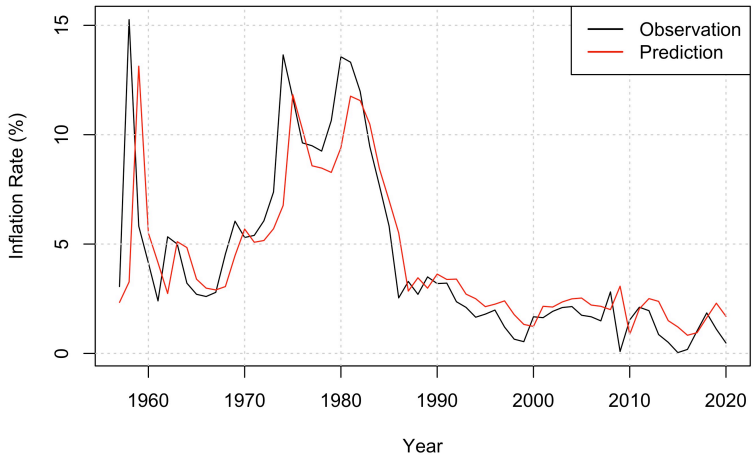


Figure 5: Observed and predicted inflation rate for the years 1957-2020 under the AR(1) model with non-varying coefficients.

2.3 Gaussian State-space Model

To obtain a more complicated model, we can specify an AR(1) model with time varying coefficients as a linear Gaussian state-space model, where for $t = 1, \ldots, T$, we have

$$\beta_t = \beta_{t-1} + V_t$$

and

$$Y_t = Y_{t-1}\beta_t + W_t \ ,$$

with $Y_0 = y_0$, $\beta_0 \sim \mathcal{N}(\mu_0, \sigma_0^2)$, $V_t \sim \mathcal{N}(0, \sigma_v^2)$, $W_t \sim \mathcal{N}(0, \sigma_w^2)$, and all $\beta_0, V_1, V_2, \ldots, V_T, W_1, W_2, \ldots, W_T$ independent. In this model, the Kalman filter can be applied to obtain the distribution of β_t conditional on $y_0, \ldots, y_t$, which is Gaussian with mean $\mu_{t|t}$ and variance $\sigma_{t|t}^2$. It is worth noting that in order use the Gaussian state space model without a constant coefficient (i.e. α), we subtract the means $\bar{y}_{1:T}$ and $\bar{y}_{0:T-1}$ from each y_t and y_{t-1}, respectively.

We first take $\sigma_v^2 = 0.01$ and $\sigma_w^2 = 4$ and obtain the distribution of the coefficient β_t for $t = 0, \ldots, T$, where $\beta_0 \sim \mathcal{N}(\mu_0, \sigma_0^2)$ (as mentioned earlier) and $\beta_t \sim \mathcal{N}(\mu_{t|t}, \sigma_{t|t}^2)$ for $t = 1, \ldots, T$ is obtained from the Kalman filter. The filtering mean and 95% credible interval are shown in Figure 6. In order to compare the predictions to those of the AR(1) model, we can also obtain the smoothing predictions, that is, the predictions $\hat{y}_t$ using $\hat{\beta}_t$, where $\hat{\beta}_t$ is conditional on all of the observations $(y_0, \ldots, y_T)$. The smoothing predictions are also shown in Figure 6 and can be compared to the predictions in Figure 5. This is an equivalent comparison because both sets of predictions are made by estimating coefficients from *all* of the observed inflation rates (over all available years).

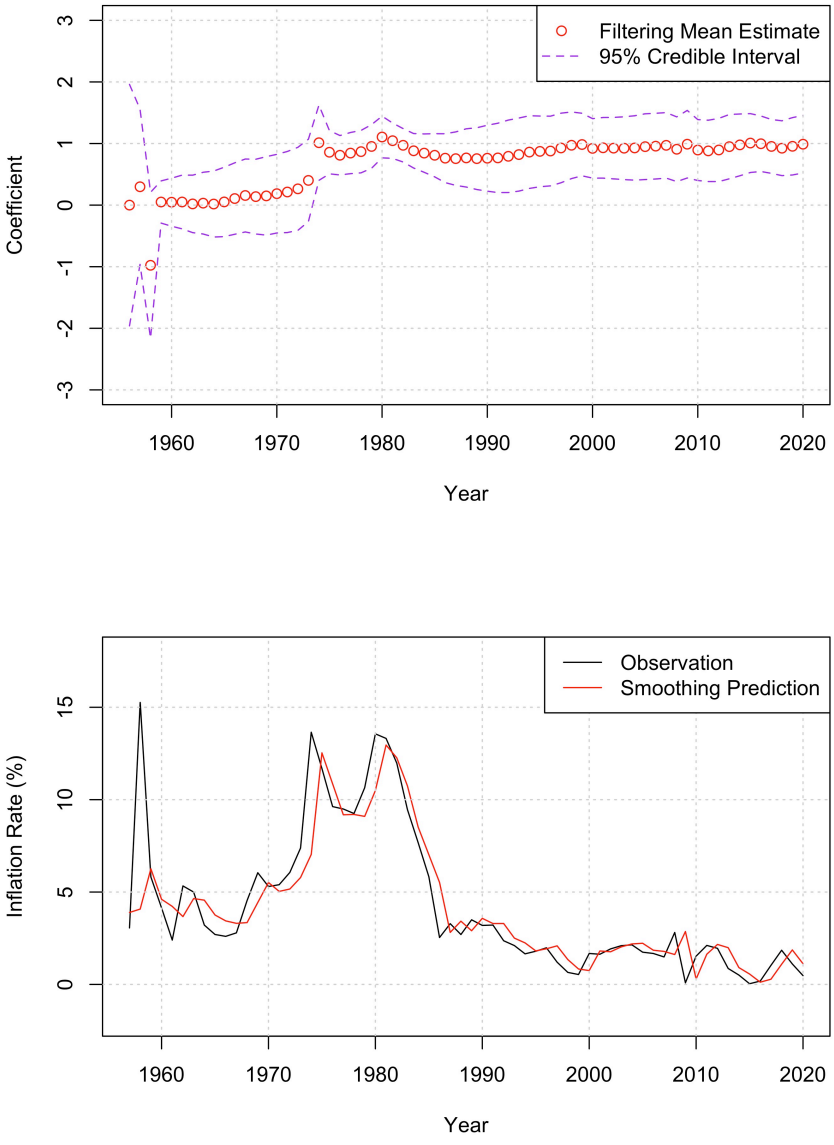


Figure 6: Top plot: filtering mean and 95% credible interval for the coefficient β_t for the years 1956-2020. Bottom plot: observed and smoothing predicted inflation rate for the years 1957-2020 under the Gaussian state-space model.

We can optimize the values of σ_v^2 and σ_w^2 by maximizing the log-likelihood – $\log p(y_1, \ldots, y_T|y_0)$ – over a grid of σ_v^2 and σ_w^2 values. The initial $\sigma_v^2 = 0.01$ and $\sigma_w^2 = 4$ correspond to a log-likelihood of -141.4971. The log-likelihood over a wide grid of σ_v^2 and σ_w^2 values is shown in Figure 7, where log-likelihoods less than -180 are not plotted. The greatest log-likelihood (-141.4979) occurred at $\sigma_v^2 = 0.01$ and $\sigma_w^2 = 4.05$.

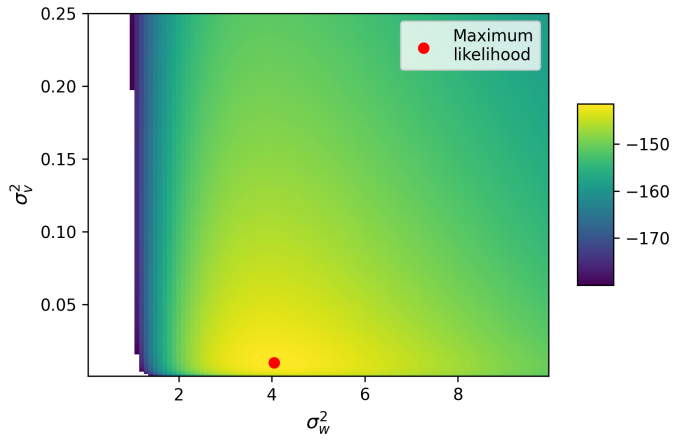


Figure 7: Log-likelihood of the observed inflation rates for the years 1957-2020 conditional on the observed inflation rate in 1956 under the Gaussian state-space model over values of σ_v^2 and σ_w^2 in increments of 0.0015 and 0.1, respectively.

We can make an additional search over σ_v^2 and σ_w^2 values more closely around the previous optimum. The log-likelihoods over this smaller grid are shown in Figures 8 and 9. A concavity is revealed around the maximum log-likelihood, which is -141.4967 at $\sigma_v^2 = 0.01016$ and $\sigma_w^2 = 4.015$.

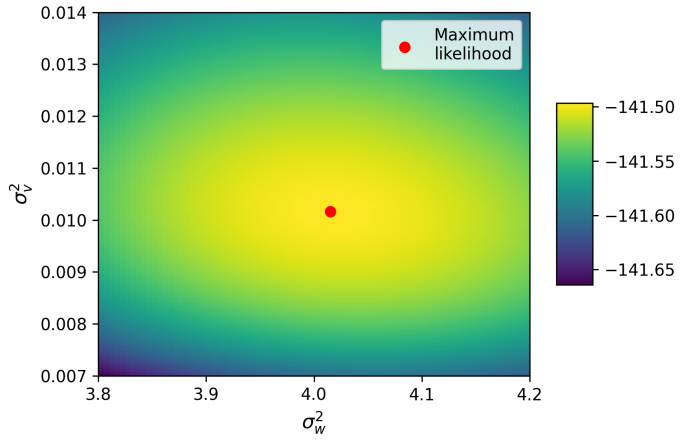


Figure 8: Log-likelihood of the observed inflation rates for the years 1957-2020 conditional on the observed inflation rate in 1956 under the Gaussian state-space model over values of σ_v^2 and σ_w^2 in increments of 0.00004 and 0.005, respectively.

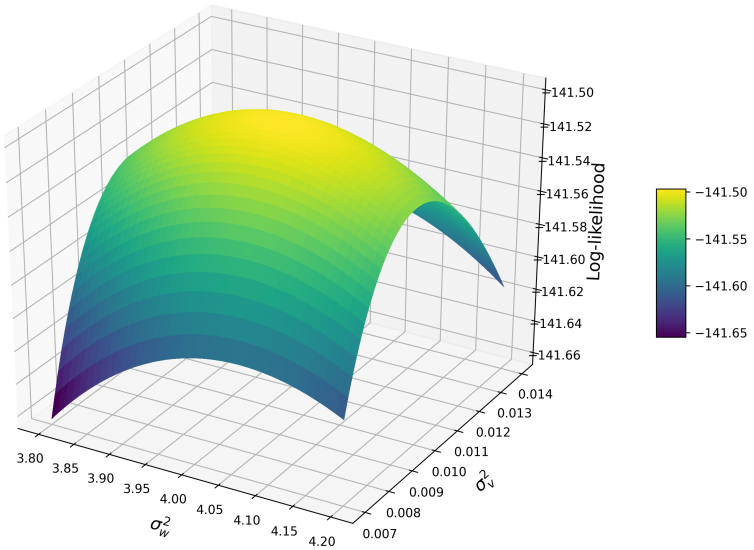


Figure 9: Histograms of the number of individuals across the ordinal (education type) and numerical variables.

Using the optimized values of σ_v^2 and σ_w^2, we can, as before, plot the Kalman filter and smoothing predictions, shown in Figure 10. A few comments can be made on the results. First, in comparison to the AR(1) model with non-varying coefficients there is no a longer positive bias in the predictions for the years 1990-2020. This is unsurprising given that the filtering mean for the coefficient approaches 1 for these years, meaning that the (filtering) expectation of the next year's inflation is simply the current year's inflation. On the other hand, for the years 1956-1970, the correlation between adjacent years for the filtering mean is around zero, which reflects the highly oscillatory inflation rate of these early years. The smoothness in the filtering mean is unsurprising as each coefficient is normal, centered on the previous coefficient and with variance σ_v^2. It is also unsurprising that there is wide uncertainty in the early years in the filtering mean, because there is little to no previous inflation data to base estimates on.

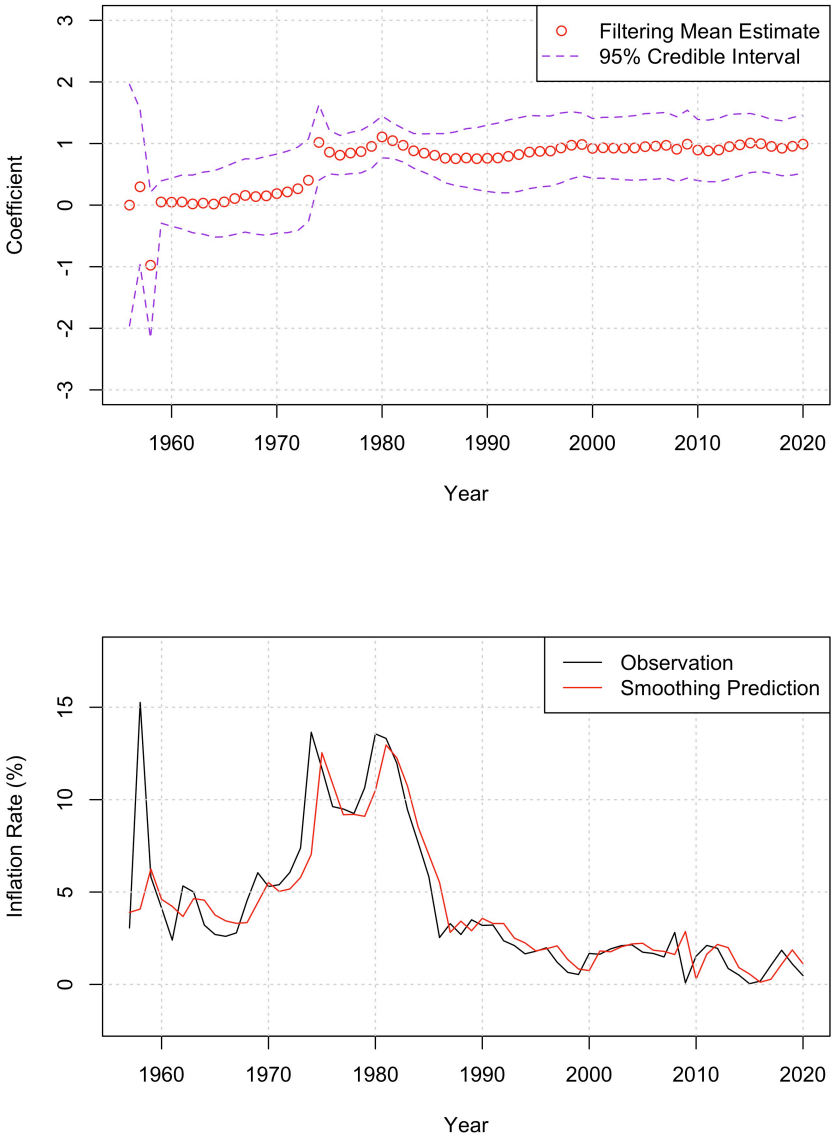


Figure 10: Histograms of the number of individuals across the ordinal (education type) and numerical variables.

2.4 Conclusions

The AR(1) showed clear problems of downward bias between years 1970-1985 and upward bias between years 1990-2020. These biases were largely rectified by using a Gaussian state-space model, which has a time-varying coefficient that is treated as the hidden state. The improvement by using this model is clear by the plots of the predictions against the observations in Figures 5 and 10 and by the empirical risk under squared loss (i.e. mean squared error), which is 5.24 for the AR(1) model and 3.66 for the Gaussian state-space model.

Exercise 1

Data

```
# Load the data
d1 <- read.csv('data/dvis.csv')

# Summarize the data
str(d1)
```

```
## 'data.frame':    1204 obs. of  11 variables:
##  $ female    : int  0 0 0 0 1 1 1 0 1 1 ...
##  $ age       : int  25 60 36 41 63 59 42 31 58 27 ...
##  $ hhninc    : num  3.1 2.4 3 3.52 0.979 2.1 4.2 5.8 1 5 ...
##  $ hhkids    : int  0 0 1 1 0 0 0 0 0 1 ...
##  $ educyrs   : num  13 10.5 18 18 9 9 12 18 9 9 ...
##  $ married   : int  0 1 0 1 0 1 1 1 1 1 ...
##  $ employed  : int  0 1 1 1 0 0 0 1 0 1 ...
##  $ docvis    : int  4 4 0 1 3 0 2 0 3 0 ...
##  $ addins    : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ eductype  : int  4 1 5 5 1 1 2 5 1 1 ...
##  $ privateins: int  0 0 0 0 0 0 1 0 0 0 ...
```

Poisson GLM and coeffficients (q 1)

```
# Define the model
m <- glm(docvis ~  age + hhninc + female + hhkids + educyrs + addins,
                   data=d1, family=poisson)
# Number of predictor variables (and the intercept)
p <- 7

# Get the general coefficient MLEs
(coefs <- coefficients(m))
```

```
##  (Intercept)          age       hhninc       female       hhkids      educyrs
##  0.364480390  0.008265545 -0.064220765  0.337526033 -0.113012241 -0.022910071
##       addins
##  0.406206655
```

```
# Save the names of the coefficients
names <- names(coefs)
```

General standard errors (q 1)

```r
# Get the general variances of the coefficients
vcov <- vcov(m)
# Get the general standard errors
se <- sqrt(diag(vcov))

# Load the testthat library, which contains `expect_equal`
library(testthat)
# Test that the calculated standard errors are equal to the R output
expect_equal(coef(summary(m))[, 2], se)
```

Robust standard errors (q 1)

```r
# Load the sandwich library, which contains `vcovHC`
library(sandwich)

# Get the robust variances of the coefficients
vcovh <- vcovHC(m)
# Get the robust standard errors
seh <- sqrt(diag(vcovh))
```

Functions

```
# Obtain the variance matrix between 'hhkids' and 'educyrs'
get_vcov_he <- function(vcov_matrix) {
  matrix(c(vcov_matrix['hhkids', 'hhkids'], vcov_matrix['hhkids', 'educyrs'], vcov_matri
x['educyrs', 'hhkids'], vcov_matrix['educyrs', 'educyrs']), nrow=2, byrow=TRUE)
}


# Wald test statistic and p-value function
get_wald <- function(var_matrix, coefs) {
  # Get the variance matrix for 'hhkids' and 'educyrs'
  v <- get_vcov_he(var_matrix)

  # Get the coefficients for 'hhkids' and 'educyrs'
  b <- coefs[c('hhkids', 'educyrs')]

  # Get the Wald test statistic
  w <- t(b) %*% solve(v) %*% b

  # Get the p-value (two hypotheses)
  p <- 1 - pchisq(w, df=2)

  # Return a list of the p-value and Wald test statistic
  return(list(p=p, w=w))
}



# Bootstrap estimation
get_boot <- function(B) {
  # Initialize the matrix for the coefficients
  coefs <- matrix(0, nrow=B, ncol=p)
  colnames(coefs) <- names

  # Initialize the Wald test statistic values
  w_values <- numeric(B)

  # Iterate through the bootstrap samples b from 1 to B
  for (b in 1:B){

    # Bootstrap sample of the data (non-random)
    set.seed(b)
    data.boot <- d1[sample(nrow(d1), replace=TRUE), ]

    # Poisson GLM for this the bootstrap sample
    m.boot <- glm(docvis ~  age + hhninc + female + hhkids + educyrs + addins,
                            data=data.boot, family=poisson)

    # Store the coefficients for this bootstrap sample
    coefs.boot <- coefficients(m.boot)
    coefs[b, ] = coefs.boot

    # Get sandwich estimate of the variances
    vcov_h.boot <- vcovHC(m.boot)
```

```
    # Get the wald test statistic
    w_values[b] <- get_wald(vcov_h.boot, coefs.boot)$w
  }

  return(list(coefs=coefs, w_values=w_values))
}
```

Bootstrap samples (q 2)

```
# Get the bootstrap object with B samples
bootstrap <- get_boot(B<-10000)

# Get the bootstrap standard errors
se_b <- sqrt(apply(bootstrap$coefs, MARGIN=2, var) * (B-1) / B)
```

Bootstrap estimates of the coefficients and the bias (q 2)

```
# Bootstrap estimate of the coefficients is the MC average
coefs_b <- apply(bootstrap$coefs, MARGIN=2, mean)

# Bootstrap bias of the coefficients
bias_coef <- coefs_b - coefs
```

Gathering the standard error results

```
se_data <- data.frame("General SE"=se,
           "Robust SE"=seh, "Bootstrap SE"=se_b,
           check.names=FALSE)
# write.csv(se_data, "exports/se_data.csv")

se_data
```

```
##              General SE   Robust SE Bootstrap SE
## (Intercept) 0.182105317 0.256141745  0.255911784
## age         0.002272531 0.003336698  0.003272877
## hhninc      0.015997210 0.020989287  0.020951691
## female      0.048737736 0.069105837  0.068472378
## hhkids      0.053270021 0.074688524  0.074742683
## educyrs     0.010740324 0.014560281  0.014559430
## addins      0.126795913 0.202007781  0.204341476
```

Gathering the coefficient results

```
# Save as a dataframe
coef_data <- data.frame("General Coefficients"=coefs,
                        "Bootstrap Coefficients"=coefs_b, "Bootstrap Bias"=bias_coef,
                        check.names = FALSE)
# write.csv(coef_data, "exports/coef_data.csv")

coef_data
```

```
##             General Coefficients Bootstrap Coefficients Bootstrap Bias
## (Intercept)          0.364480390            0.364652697   1.723075e-04
## age                  0.008265545            0.008261299  -4.245935e-06
## hhninc              -0.064220765           -0.064907776  -6.870116e-04
## female               0.337526033            0.338130453   6.044199e-04
## hhkids              -0.113012241           -0.113681281  -6.690406e-04
## educyrs             -0.022910071           -0.022997735  -8.766455e-05
## addins               0.406206655            0.385118326  -2.108833e-02
```

Variance covariance matrix of the bootstrap coefficients (q 3)

```
# Center the bootstrap coefficients to their means
C <- sweep(bootstrap$coefs, 2, apply(bootstrap$coefs, 2, mean), "-")

# Obtain the population covariance matrix
vcov_b <- t(C) %*% C / B

# Test equality with the previous bootstrap standard error calculations
library(testthat)
expect_equal(se_b, sqrt(diag(vcov_b)))
```

Wald test and p-values (q 3)

```
# General Wald
w <- get_wald(vcov, coefs)

# Robust Wald
wh <- get_wald(vcovh, coefs)

# Bootstrap Wald
w_b <- get_wald(vcov_b, coefs_b)
```

Gathering the Wald test results (q 3)

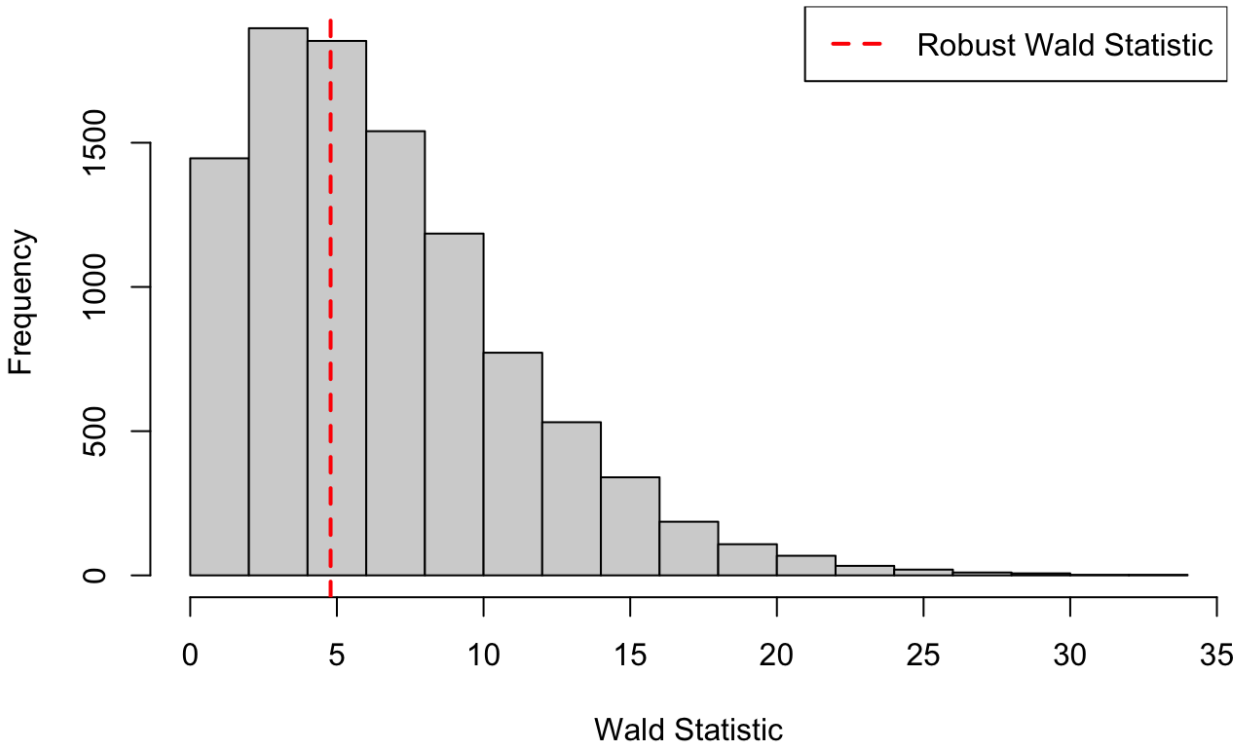
```
wald_data <- data.frame("General"=c(w$w, w$p), "Robust"=c(wh$w, wh$p),
                        "Bootstrap"=c(w_b$w, w_b$p))
rownames(wald_data) <- c("Wald Statistic", "p-value")

wald_data
```

```
##                    General     Robust Bootstrap
## Wald Statistic 8.70927258 4.78588359 4.7257278
## p-value        0.01284711 0.09136052 0.0941502
```

```
# write.csv(wald_data, "exports/wald_data.csv")
```

Plot the bootstrap Wald test statistics

```
# jpeg("plots/boot_hist.jpg", width=6.4, height=4.8, units="in", res=300)
hist(bootstrap$w_values, main=NULL, xlab="Wald Statistic")
abline(v=wh$w, col="red", lwd=2, lty=2)
legend("topright", c("Robust Wald Statistic"), lty=2, lwd=2, col="red")
```




```
# dev.off()
```

Get the bootstrap p-value (q 4)

```
# Get the robust wald test statistic
(wald_robust <- wh$w[1])
```

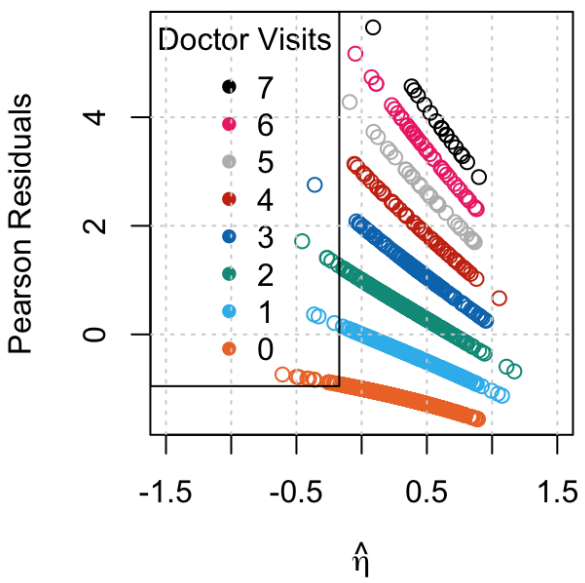
```
## [1] 4.785884
```

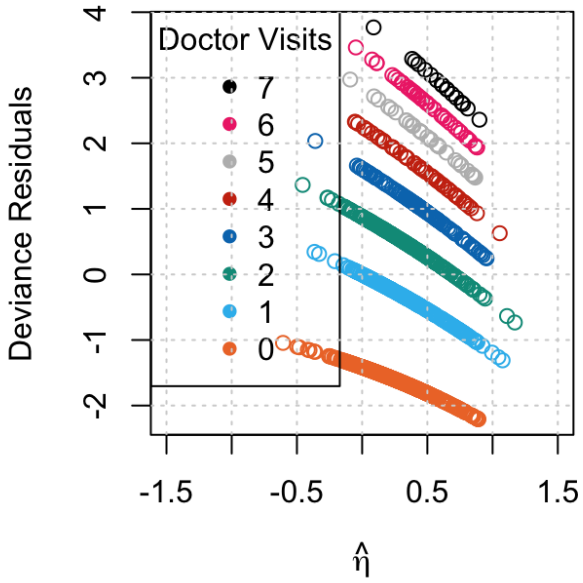
```
# Get the bootstrap p-value
(p_boot <- mean(bootstrap$w_values > wald_robust))
```

```
## [1] 0.5922
```

Plot residuals (q 5)

```
## Set colors
cblue <- "#0077BB"
ccyan <- "#33BBEE"
cteal <- "#009988"
corange <- "#EE7733"
cred <- "#CC3311"
cmagenta <- "#EE3377"
cgrey <- "#BBBBBB"
cblack <- "#000000"
ccol <- c(corange, ccyan, cteal, cblue, cred, cgrey, cmagenta, cblack)

# Linear predictor
eta_hat <- predict(m, type='link')

# Pearson residuals
r_p <- residuals(m, type="pearson")
# Deviance residuals
r_d <- residuals(m, type="deviance")

# Number of doctor visits
num_visits <- as.character(c(0:7))

# jpeg("plots/residuals.jpg", width=8*1.2, height=6*1.2, units="in", res=300)
par(mfrow=c(1, 2),  pty="s")
plot(eta_hat, r_p, col=ccol[d1$docvis+1], xlim=c(-1.5, 1.5),
     ylab='Pearson Residuals', xlab=expression(hat(eta)))
legend("topleft", rev(num_visits), title="Doctor Visits",
       col = rev(ccol), pch = 16, bg="transparent")
grid()

plot(eta_hat, r_d, col=ccol[d1$docvis+1], xlim=c(-1.5, 1.5),
     ylab='Deviance Residuals', xlab=expression(hat(eta)))
legend("topleft", rev(num_visits), title="Doctor Visits",
       col = rev(ccol), pch = 16, bg="transparent")
grid()
```


Pearson Residuals
Doctor Visits
7
6
5
4
3
2
1
0
0
2
4
-1.5
-0.5
0.5
1.5
$\hat{\eta}$


Deviance Residuals
Doctor Visits
7
6
5
4
3
2
1
0
-2
-1
0
1
2
3
4
-1.5
-0.5
0.5
1.5
$\hat{\eta}$

```
# dev.off()
```

Exercise 2

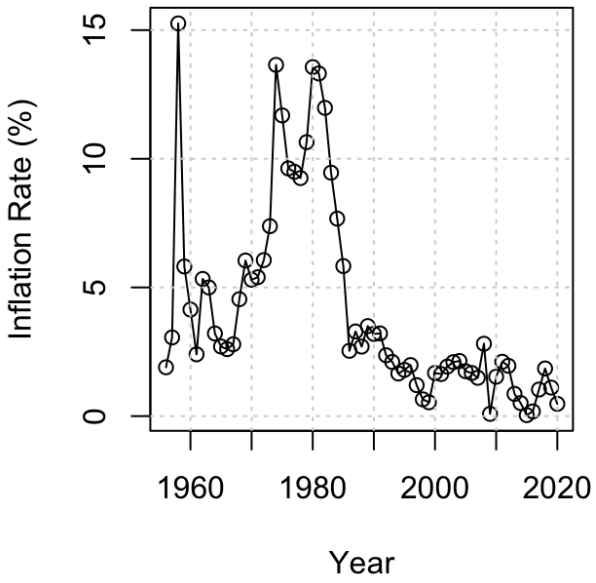
Data

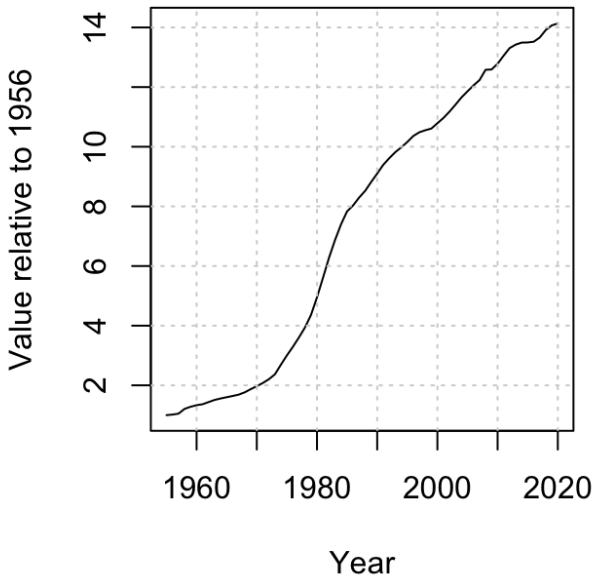
```
# Load the data
d2 <- read.csv('data/infl.csv')

# Summarize the data
str(d2)
```

```
## 'data.frame':    65 obs. of  2 variables:
##  $ year  : int  1956 1957 1958 1959 1960 1961 1962 1963 1964 1965 ...
##  $ inflFR: num  1.9 3.06 15.26 5.82 4.14 ...
```

Exploratory data analysis (q 1)

```
# jpeg("plots/inflation.jpg", width=10, height=8, units="in", res=400)
par(mfrow=c(1, 2), pty='s')
plot(d2, type='o', xlab="Year", ylab="Inflation Rate (%)")
grid()

plot(c(1955, d2$year), c(1, cumprod(1 + d2$inflFR / 100)), type='l',
     xlab="Year", ylab="Value relative to 1956")
grid()
```






```
# dev.off()
```

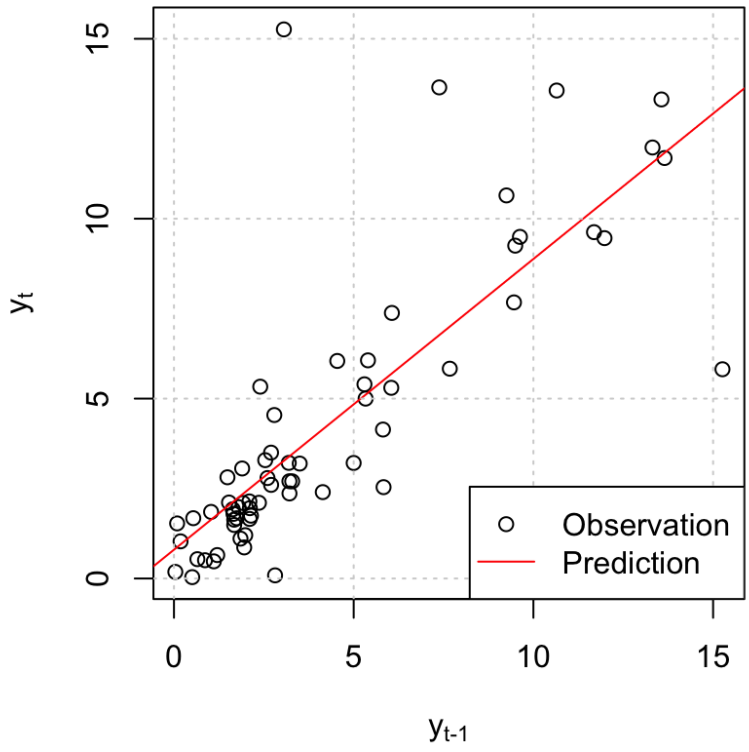
Set up AR(1) model (q 2)

```
# The number of years for all the inflation data
T1 <- length(d2$inflFR )

# Inflation data for 1:T
infl <- as.matrix(d2$inflFR [2:T1])

# Inflation data for 0:T-1
laginfl <- as.matrix(d2$inflFR [1:T1-1])

# Set `year` to be the years for 1:T
year <- as.matrix(d2$year[2:T1])

# Fit a linear model
m2 <- lm(infl ~ laginfl)

summary(m2)
```

```
##
## Call:
## lm(formula = infl ~ laginfl)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.3170 -0.8817 -0.3684  0.4601 11.9893
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.80032    0.43594   1.836   0.0712 .
## laginfl      0.80809    0.07581  10.660 1.19e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.325 on 62 degrees of freedom
## Multiple R-squared:  0.647,  Adjusted R-squared:  0.6413
## F-statistic: 113.6 on 1 and 62 DF,  p-value: 1.193e-15
```

Plot a scatter plot showing the linear AR(1) model (q 2)

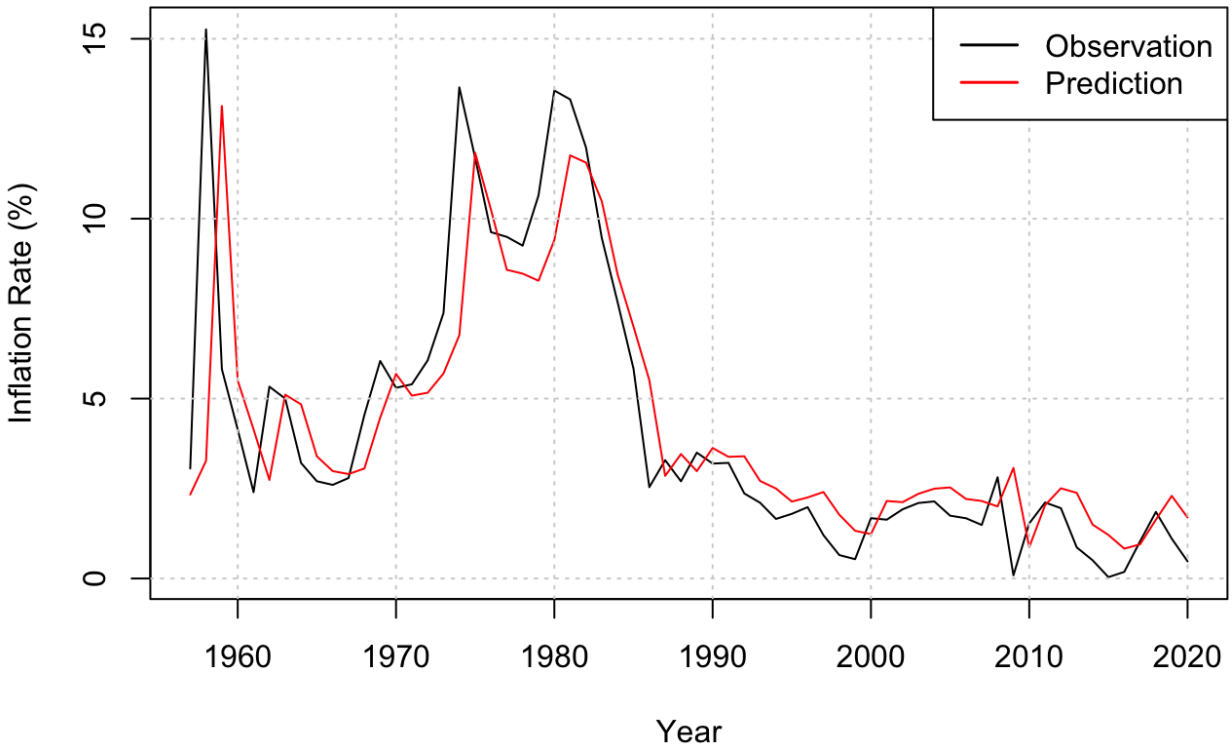
```
# jpeg("plots/scatter.jpg", width=6, height=6, units="in", res=400)
par(mfrow=c(1, 1), pty="s")
plot(laginfl, infl, xlab=expression("y"["t-1"]),
     ylab=expression("y"["t"]))
abline(a=coef(m2)[1], b=coef(m2)[2], col="red")
grid()
legend("bottomright", c("Observation", "Prediction"),
       pch=c(1, NA), lty=c(NA, 1), col=c("black", "red"))
```




```
# dev.off()
```

Plot of AR(1) model predictions (q 2)

```
# jpeg("plots/ar1.jpg", width=7, height=5, units = "in", res = 400)
plot(year, infl, xlab="Year", ylab="Inflation Rate (%)", type="l")
lines(year, predict(m2), col="red", main="Prediction", type="l")
grid()
legend("topright", c("Observation", "Prediction"),
       col=c("black", "red"), lwd=1.5)
```




```
# dev.off()
```

Kalman functions

```
kalman = function(y, F, G, Q, H, R, mu0, Sigma0) {
  dy = nrow(y)
  T = ncol(y)
  dx = length(mu0)
  I = diag(dx)

  ## INITIALIZATION ##
  mu.p = matrix(0, nrow = dx, ncol = T)
  Sigma.p = array(0, c(dx, dx, T))
  mu.f = matrix(0, nrow = dx, ncol = T)
  Sigma.f = array(0, c(dx, dx, T))
  mu.s = matrix(0, nrow = dx, ncol = T)
  Sigma.s = array(0, c(dx, dx, T))

  ## FORWARD RECURSION ## Time 1
  mu.p[, 1] = F %*% mu0
  Sigma.p[, , 1] = F %*% Sigma0 %*% t(F) + G %*% Q %*% t(G)

  nu = y[, 1] - H[1] %*% mu.p[, 1]
  # Time varying H
  S = H[1] %*% Sigma.p[, , 1] %*% t(H[1]) + R
  K = Sigma.p[, , 1] %*% t(H[1]) %*% solve(S)
  mu.f[, 1] = mu.p[, 1] + K %*% nu
  Sigma.f[, , 1] = (I - K %*% H[1]) %*% Sigma.p[, , 1]

  # Time 2:T
  for (t in (2:T)) {
    # Prediction
    mu.p[, t] = F %*% mu.f[, t - 1]
    Sigma.p[, , t] = F %*% Sigma.f[, , t - 1] %*% t(F) + G %*% Q %*% t(G)

    # Update
    nu = y[, t] - H[t] %*% mu.p[, t]
    S = H[t] %*% Sigma.p[, , t] %*% t(H[t]) + R
    K = Sigma.p[, , t] %*% t(H[t]) %*% solve(S)
    mu.f[, t] = mu.p[, t] + K %*% nu
    Sigma.f[, , t] = (I - K %*% H[t]) %*% Sigma.p[, , t]
  }

  ## BACKWARD RECURSION ##
  mu.s[, T] = mu.f[, T]
  Sigma.s[, , T] = Sigma.f[, , T]
  for (t in (T - 1):1) {
          J = Sigma.f[, , t] %*% t(F) %*% solve(Sigma.p[, , t + 1])
          mu.s[, t] = mu.f[, t] + J %*% (mu.s[, t + 1] - mu.p[, t + 1])
          Sigma.s[, , t] = Sigma.f[, , t] + J %*% (Sigma.s[, , t + 1] - Sigma.p[,
              , t + 1]) %*% t(J)
  }

  ## LOG-LIKELIHOOD ##

  # Initialize the probabilities
  probs = numeric(T)
```

```
  # Special case for t = 1
  mu.t = H[1] %*% mu0
  Sigma.t = H[1] %*% Sigma.p[, , 1] %*% t(H[1]) + R
  probs[1] = dnorm(y[, 1], mean=mu.t, sd=sqrt(Sigma.t))

  # Iterate over t = 2:T
  for (t in (2:T)) {
    mu.t = H[t] %*% mu.p[, t]
    Sigma.t = H[t] %*% Sigma.p[, , t] %*% t(H[t]) + R
    probs[t] = dnorm(y[, t], mean=mu.t, sd=sqrt(Sigma.t))
  }

  # Sum of the logs
  log_like <- sum(log(probs))

  return(list(mu.f = mu.f, Sigma.f = Sigma.f, mu.p = mu.p, Sigma.p = Sigma.p,
          mu.s = mu.s, Sigma.s = Sigma.s, log_like = log_like))
}


# Function to store relevant information from the Kalman results
get_kf_info <- function(kf_results, alpha=0.05) {

  # Filtering mean vector and variance matrix for hidden states 1:T
  mu.f = kf_results$mu.f
  Sigma.f = kf_results$Sigma.f

  # Smoothing mean vector for hidden states 1:T
  mu.s = kf_results$mu.s

  # Get the standard error for the hidden states
  se.f = sqrt(Sigma.f[,,])

  # Years 0:T (requires d2 dataframe)
  year0 <- d2$year

  # Mean vector and variance matrix for hidden states 0:T
  mu.f0 = c(mu0,mu.f)
  se.f0 = c(sqrt(Sigma0),se.f)

  # Get confidence intervals for the hidden states
  cv = qnorm(1-alpha/2)
  # From 1:T
  CIupper = mu.f+cv*se.f
  CIlower = mu.f-cv*se.f
  # From 0:T
  CIupper0 = mu.f0+cv*se.f0
  CIlower0 = mu.f0-cv*se.f0

  return(list(mu.f=mu.f, Sigma.f=Sigma.f, se.f=se.f, mu.s=mu.s, year0=year0,
              mu.f0=mu.f0, se.f0=se.f0, CIupper=CIupper, CIlower=CIlower,
              CIupper0=CIupper0, CIlower0=CIlower0))
}
```

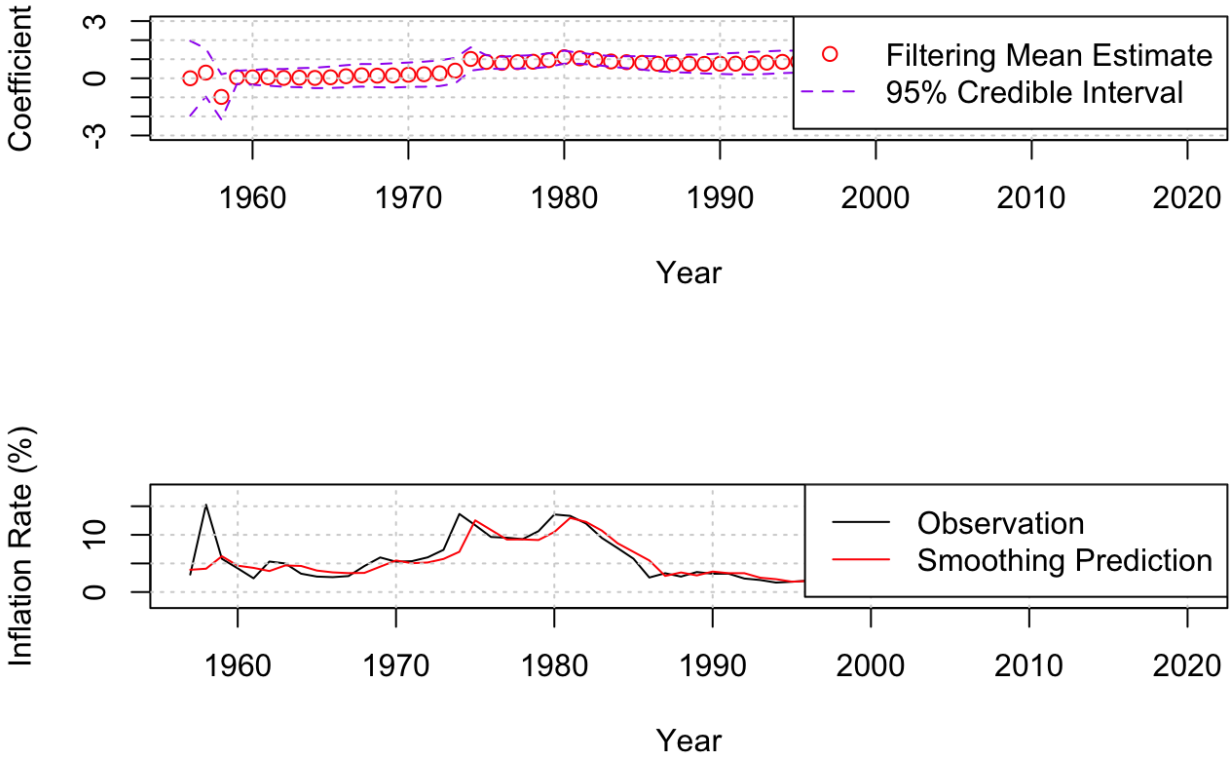
```
# Prediction of y from beta (used for smoothing mean predictions)
get_y <- function(beta) {
  # Add back the mean of y_{1:T} to get the inflation rate prediction
  return((t(H) * beta) + mean(infl))
}


# Plot of beta estimates and smoothing predictions (q 3a)
plot_kalman <- function(kf_results, alpha=0.05) {
  par(mfrow=c(2, 1))

  kf_info <- get_kf_info(kf_results, alpha)

  plot(kf_info$year0, kf_info$mu.f0, col="red", ylim=c(-3, 3),
       ylab="Coefficient", xlab="Year")
  lines(kf_info$year0, kf_info$CIupper0, col="purple", lty=2)
  lines(kf_info$year0, kf_info$CIlower0, col="purple", lty=2)
  grid()
  legend("topright", c("Filtering Mean Estimate", "95% Credible Interval"),
       col=c("red", "purple"), lty=c(NA, 2), pch=c(1, NA))

  plot(year, infl, type="l", ylim=c(-2, 18),
       ylab="Inflation Rate (%)", xlab="Year")
  lines(year, get_y(kf_info$mu.s), col="red")
  grid()
  legend("topright", c("Observation", "Smoothing Prediction"),
       col=c("black", "red"), lty=c(1, 1))
}
```

Initialize the Kalman filter (q 3a)

```
# y values (row vector)
y <- t(infl-mean(infl))

# The H_t values
H <- laginfl-mean(laginfl)

# T (number of years)
T <- length(year)

# Initialization of the (hidden) coefficients
mu0 <- 0
Sigma0 <- 1

# State transition and noise transfer
F <- 1
G <- 1

# Variances (specific to q 3a)
R <- 4
Q <- 0.01
```

Get and plot the Kalman results for the (base) estimates of R and Q (q 3a)

```
# Get the Kalman results using the initial (base) R and Q
results_base = kalman(y, F, G, Q, H, R, mu0, Sigma0)

# jpeg("plots/results_base.jpg", width=7, height=10, units="in", res = 400)
plot_kalman(results_base)
```




```
# dev.off()
```

Base log-likelihood (q 3b)

```
results_base$log_like
```

```
## [1] -141.4971
```

Grid search function (q 3b)

```
# Obtain a dataframe of R and Q values, and the corresponding log-likelihood
grid_search <- function(R_range, Q_range) {
  # Initially R = 4 and Q = 0.01

  # Obtain the grid of R and Q values as a dataframe
  dg <- expand.grid(R_range, Q_range)
  colnames(dg) <- c("R_range","Q_range")

  # Initialize the log-likelihood column
  n_grid <- nrow(dg)
  dg[,"log_like"] = numeric(n_grid)

  for (i in (1:n_grid)) {

    # Store the R and Q values
    R <- dg[i, ]$R_range
    Q <- dg[i, ]$Q_range

    # Get the Kalman filter
    rkf = kalman(y, F, G, Q, H, R, mu0, Sigma0)

    # Store the likelihood
    dg[i, "log_like"] <- rkf$log_like
  }

  # Return the dataframe and the maximum log-likelihood of it
  return(list(dg=dg, dg_max=dg[dg$log_like == max(dg[,"log_like"]), ]))
}
```

Base grid (q 3b)

```
# Get the base grid search
grid_base <- grid_search(R_range=seq(0.05, 10, 0.1), Q_range=seq(0.001, 0.25, 0.0015))

# Store the base grid search
write.csv(grid_base$dg, "exports/grid_base.csv")
```

Maximum base log-likelihood (q 3b)

```
grid_base$dg_max
```

```
##     R_range Q_range  log_like
## 641    4.05    0.01 -141.4979
```

Fine grid (q 3b)

```
# Get the fine grid search
grid_fine <- grid_search(R_range=seq(3.8, 4.2, 0.005), Q_range=seq(0.007, 0.014, 0.00004
))

# Store the fine grid search
write.csv(grid_fine$dg, "exports/grid_fine.csv")
```

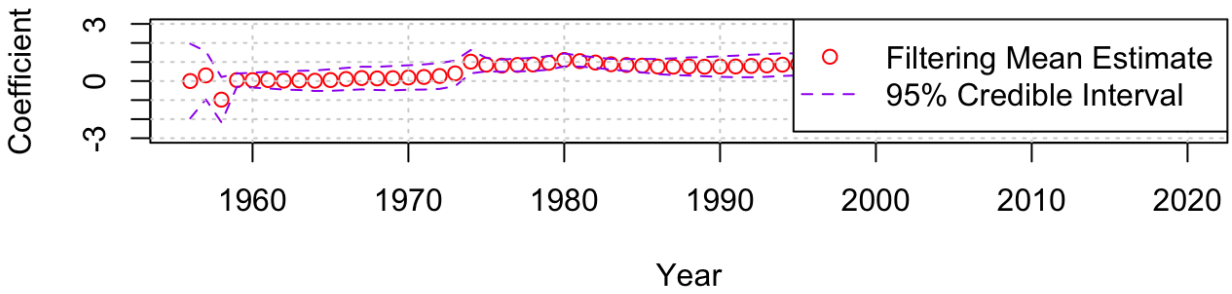
Maximum fine log-likelihood (q 3b)

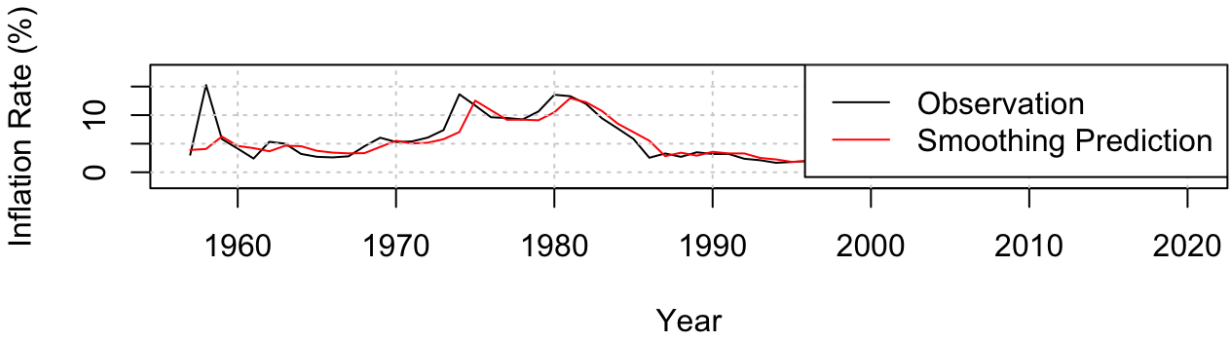
```
grid_fine$dg_max
```

```
##      R_range Q_range  log_like
## 6443   4.015 0.01016 -141.4967
```

Get and plot the Kalman results for the (optimized) estimates of R and Q (q 3c)

```
results_fine = kalman(y, F, G, 0.01014, H, 4.015, mu0, Sigma0)

# jpeg("plots/results_fine.jpg", width=7, height=10, units="in", res = 400)
plot_kalman(results_fine)
```






```
# dev.off()
```

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [2]:
```
save = True
```

In [3]:
```
df1 = pd.read_csv("exports/grid_base.csv", index_col=0)
df2 = pd.read_csv("exports/grid_fine.csv", index_col=0)
```

In [4]:
```
df1.head()
```

Out[4]:

	R_range	Q_range	log_like
1	0.05	0.001	-inf
2	0.15	0.001	-796.711600
3	0.25	0.001	-513.551962
4	0.35	0.001	-391.798832
5	0.45	0.001	-324.659364

In [5]:
```
df2.head()
```

Out[5]:

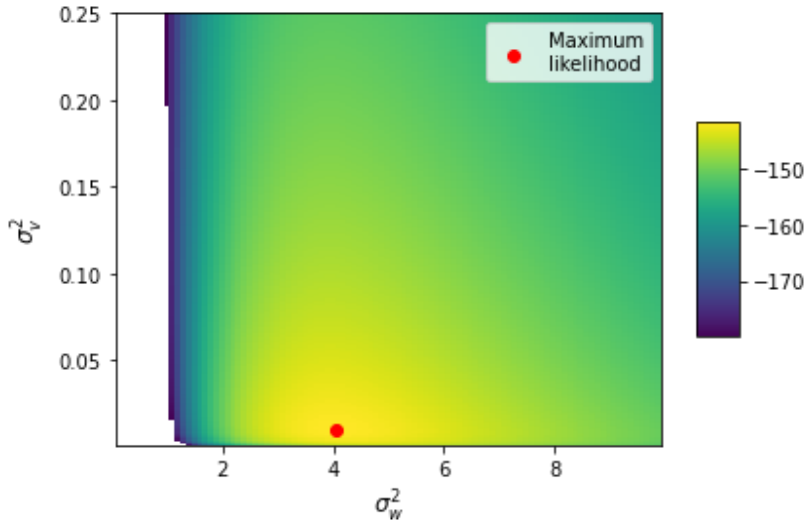
	R_range	Q_range	log_like
1	3.800	0.007	-141.664072
2	3.805	0.007	-141.661518
3	3.810	0.007	-141.659019
4	3.815	0.007	-141.656576
5	3.820	0.007	-141.654188

In [6]:

```
def get_mesh(df, threshold=-180):

    # Initialize the meshgrid
    rv, qv = np.meshgrid(df.R_range.unique(), df.Q_range.unique())

    # Initialize the shape
    assert rv.shape == qv.shape
    s = rv.shape

    # Initialize the log-likelihoods
    zv = np.zeros(s)

    # Loop through the R and Q range, storing the log-likelihoods
    for i in range(s[0]):
        for j in range(s[1]):
            log_like = df.loc[(df.R_range == rv[i, j]) & (
                              df.Q_range == qv[i, j])
                              ].log_like.values[0]

            # Apply the threshold
            if log_like < threshold:
                zv[i, j] = np.nan
            else:
                zv[i, j] = log_like

    return rv, qv, zv


def plot_mesh(mesh):
    fig = plt.figure()
    ax = fig.gca()

    # Create a surface plot
    surf = ax.pcolormesh(*mesh, linewidth=0, antialiased=False)

    # Add a colorbar
    fig.colorbar(surf, shrink=0.5, aspect=5)

    # Set the labels
    label_fontsize = 12
    ax.set_xlabel("$\sigma^2_w$", fontsize=label_fontsize)
    ax.set_ylabel("$\sigma^2_v$", fontsize=label_fontsize)

    # Get the maximum likelihood
    rv, qv, zv = mesh
    arg_max = np.where(zv == np.nanmax(zv))
    # Plot as a point
    ax.scatter(rv[arg_max][0], qv[arg_max][0],
               color="red", label='Maximum\nlikelihood')

    # Apply legend
    ax.legend()
    plt.tight_layout()
```

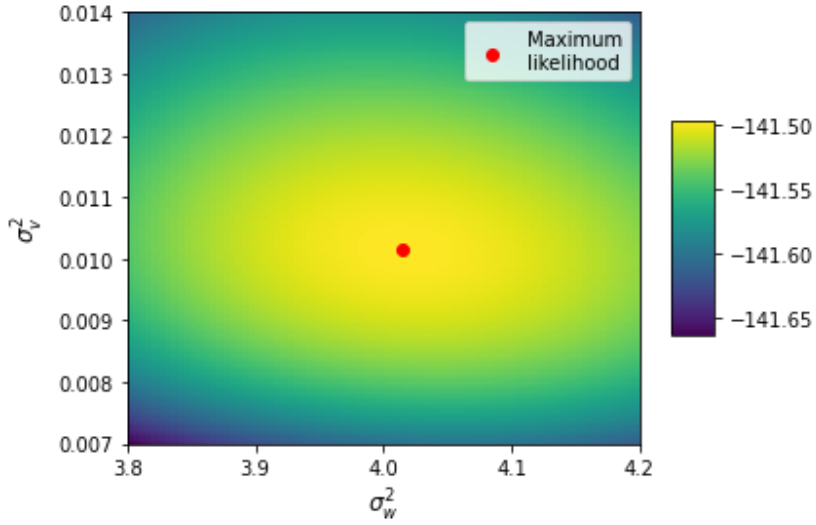
In [7]:
```
# Get the mesh for the base grid search
mesh1 = get_mesh(df1)
```

In [8]:
```
plot_mesh(mesh1)
if save:
    plt.savefig('plots/mesh1.png', dpi=300)
```

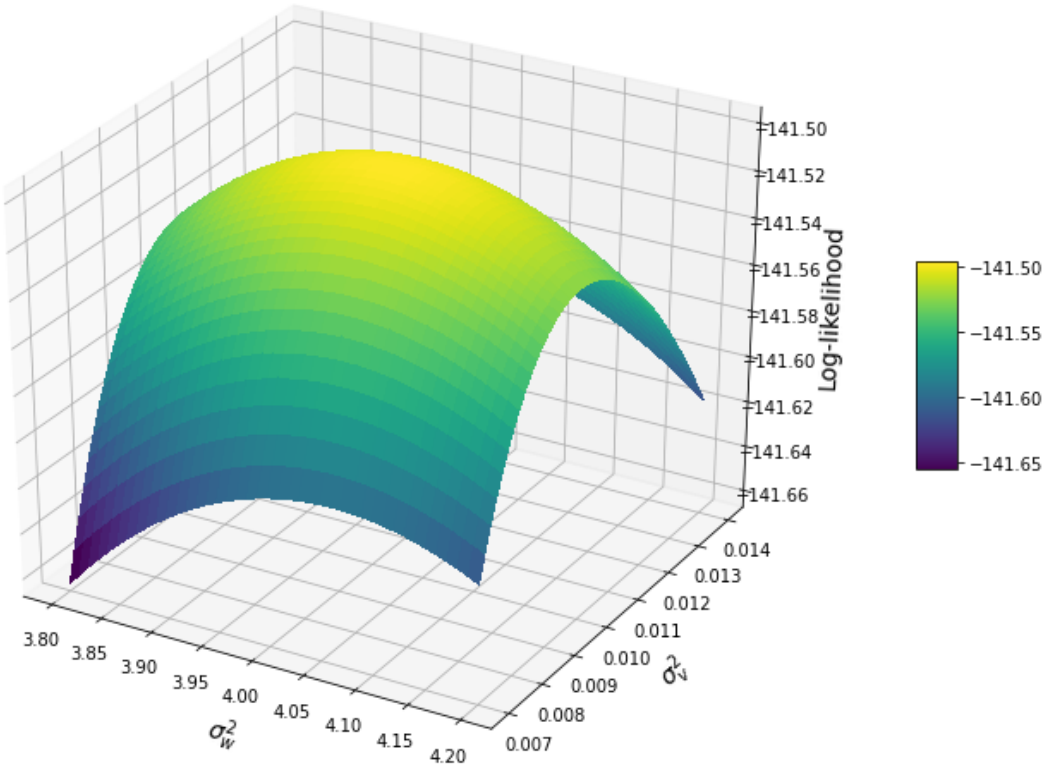


In [9]:
```
# Get the mesh for the fine grid search
mesh2 = get_mesh(df2)
```

In [10]:
```
plot_mesh(mesh2)
if save:
    plt.savefig('plots/mesh2.png', dpi=300)
```



In [14]:
```
from matplotlib import cm

fig, ax = plt.subplots(subplot_kw={"projection": "3d"},
                       figsize=(10, 7))

# Plot the surface
surf = ax.plot_surface(*mesh2, cmap=cm.viridis,
                       linewidth=0, antialiased=False)

# Label the axes
label_fontsize = 14
ax.set_xlabel("\n$\sigma^2_w$", fontsize=label_fontsize)
ax.set_ylabel("\n$\sigma^2_v$", fontsize=label_fontsize)
ax.set_zlabel("\nLog-likelihood", fontsize=label_fontsize)

# Add a colorbar
fig.colorbar(surf, shrink=0.25, aspect=5)

plt.tight_layout()
if save:
    plt.savefig('plots/surface2.png', dpi=300)
plt.show()
```



In [ ]: