Esercitazione 6

November 14, 2024

Prendiamo il dataset che contiene, per alcune stazione Europee, diverse misure ambientali

[22]:
```
library(ggplot2)
library(tidyverse)
library(magrittr)
library(gridExtra)
setwd("/Users/gianlucamastrantonio/Dropbox (Politecnico di Torino Staff)/
 ↪Didattica/statistica computazionale/esercizi")
load("dataset_clima_long.RData")
summary(dataset_clima)
dataset_clima <- as.data.frame(dataset_clima)
```

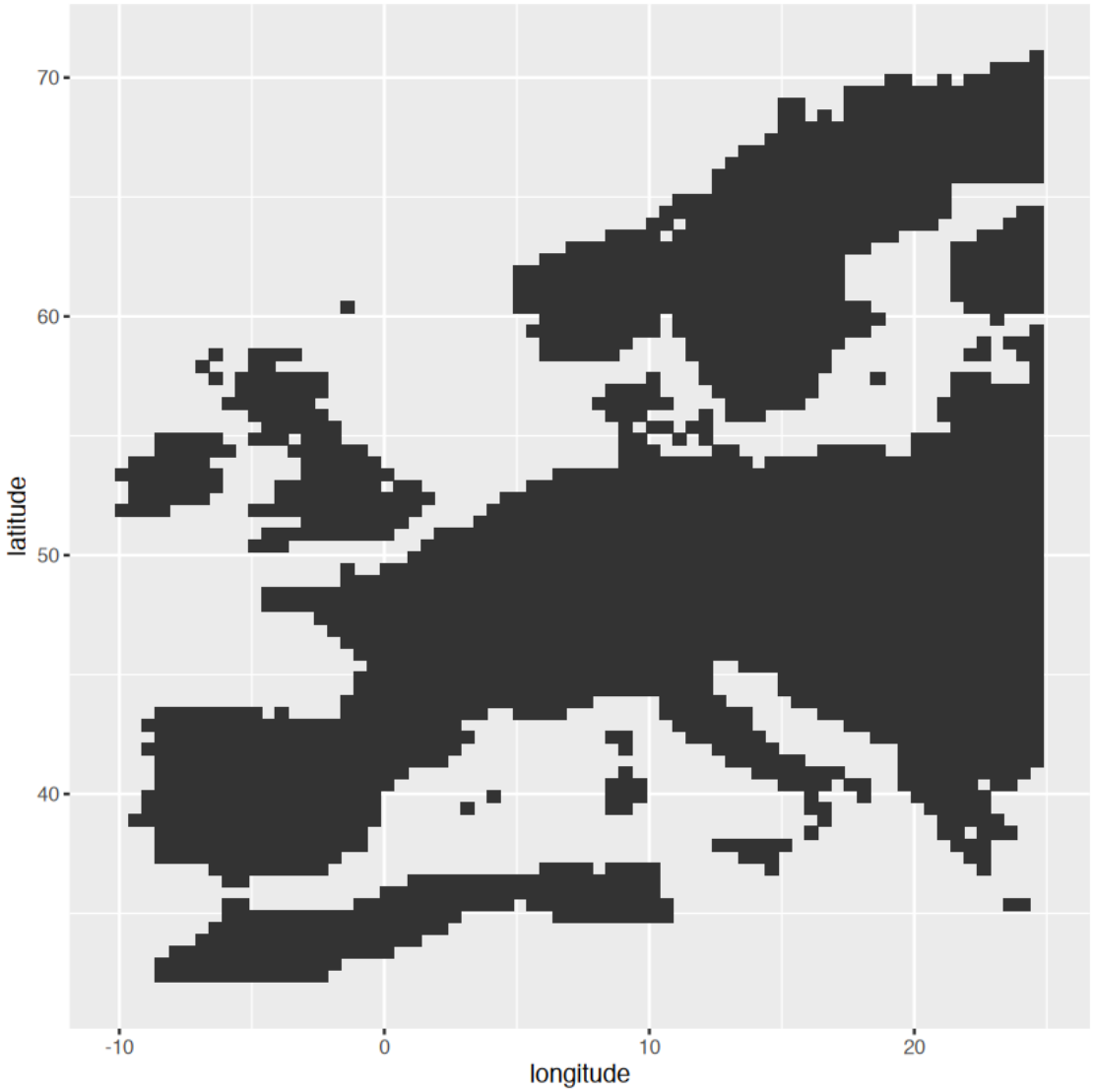
```
sea_level_pressure mean_global_radiation precipitation_sum mean_temperature
Min.   : 821.2     Min.   :   0.0        Min.   :  0.0     Min.   :-40.36
1st Qu.:1009.9     1st Qu.:  51.0        1st Qu.:  0.0     1st Qu.:  3.70
Median :1015.5     Median : 124.0        Median :  0.0     Median :  9.86
Mean   :1014.9     Mean   : 135.1        Mean   :  2.1     Mean   :  9.44
3rd Qu.:1020.9     3rd Qu.: 210.0        3rd Qu.:  2.2     3rd Qu.: 15.70
Max.   :1069.0     Max.   :2777.0        Max.   :228.5     Max.   : 35.79
NA's   :227204     NA's   :357129        NA's   :270469    NA's   :120845
minimum_temperature maximum_temperature    humidity         longitude
Min.   :-41.52      Min.   :-39.14      Min.   :12.1      Min.   :-9.875
1st Qu.:  0.20      1st Qu.:  7.22      1st Qu.:68.2      1st Qu.: 2.125
Median :  5.73      Median : 14.29      Median :78.0      Median :11.625
Mean   :  5.22      Mean   : 14.02      Mean   :75.6      Mean   :10.344
3rd Qu.: 11.04      3rd Qu.: 21.08      3rd Qu.:85.3      3rd Qu.:18.625
Max.   : 30.49      Max.   : 46.14      Max.   :94.5      Max.   :24.625
NA's   :65821       NA's   :55112       NA's   :1142233
   latitude          time         external
Min.   :32.38   Min.   :   0   Min.   :  24
1st Qu.:43.38   1st Qu.: 912   1st Qu.:3650
Median :49.38   Median :1824   Median :3650
Mean   :50.36   Mean   :1824   Mean   :3602
3rd Qu.:56.88   3rd Qu.:2737   3rd Qu.:3650
Max.   :70.88   Max.   :3649   Max.   :3650
```

Vediamo la prima delle stazioni, facendo attenzione che i dati sono organizzati in modo che le prime 2496 righe sono i punti al tempo 1 di tutte le stazioni, le secondo 2496 righe sono i secondi punti

temporali, etc

[23]:
```
nstaz <- 2496
ntempi <- dim(dataset_clima)[1]/nstaz
ntempi
dataset_clima %>%
  select(longitude, latitude) %>%
  slice(1:nstaz)%>%
  ggplot(aes(x = longitude, y = latitude)) + geom_tile()
```

3650



Giusto per testare un po' ggplot, vediamo come possiamo calcolare qualche statistica divisa per stazione, per esempio la variabilità e la media delle variabili

[ ]:

[ ]:

[24]:
```
summary_stat <- dataset_clima %>%
  mutate(station_id = rep(1:nstaz, times = ntempi)) %>%
  group_by(station_id) %>%
  summarize(across(
    c(
      sea_level_pressure, mean_global_radiation, precipitation_sum,
      mean_temperature, minimum_temperature, maximum_temperature,
      humidity
    ),
    list(
      mean = ~ mean(.x, na.rm = TRUE),
      variance = ~ var(.x, na.rm = TRUE)
    ),
    .names = "{.col}_{.fn}"
  ))%>%
  mutate(
    longitude = dataset_clima$longitude[1:nstaz],
    latitude = dataset_clima$latitude[1:nstaz]
    )

head(summary_stat)
```

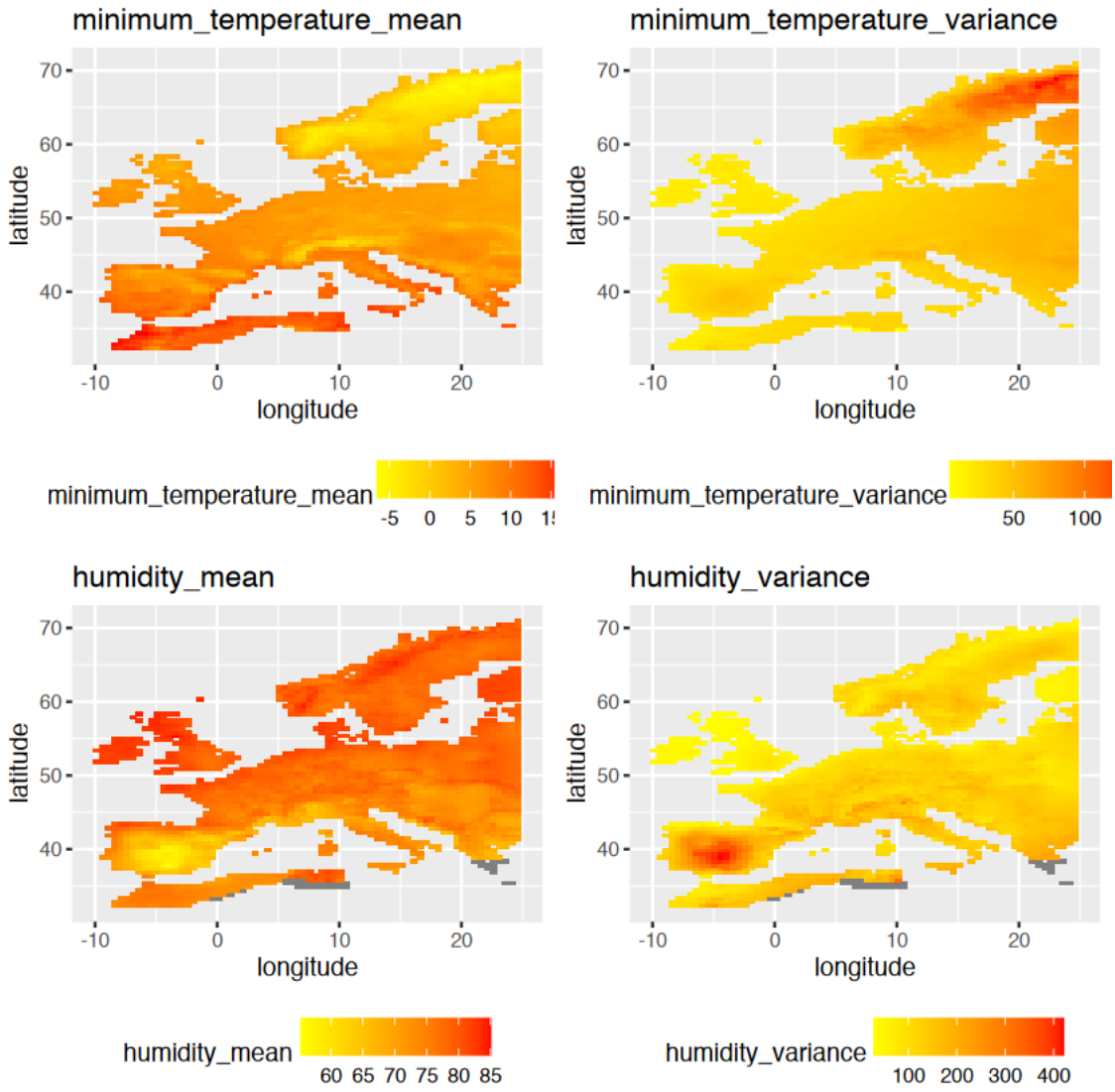
A tibble: 6 x 17

station_id <int>	sea_level_pressure_mean <dbl>	sea_level_pressure_variance <dbl>	mean_global_radiation <dbl>
1	1017.248	24.66679	NaN
2	1017.171	25.35685	NaN
3	1017.751	27.17540	NaN
4	1017.555	27.48265	232.8591
5	1017.480	27.90072	233.7127
6	1017.403	28.29994	230.7155

[25]:
```
p1 <- summary_stat %>%
  ggplot(aes(x = longitude, y = latitude)) +
  geom_tile(aes(fill = minimum_temperature_mean)) +
  theme(legend.position = "bottom") +
  scale_fill_gradient(low = "yellow",high = "red") +␣
 ↪ggtitle("minimum_temperature_mean")

p2 <- summary_stat %>%
  ggplot(aes(x = longitude, y = latitude)) +
  geom_tile(aes(fill = minimum_temperature_variance)) +
  theme(legend.position = "bottom") +
```

```
  scale_fill_gradient(low = "yellow", high = "red") +␣
 ↪ggtitle("minimum_temperature_variance")

p3 <- summary_stat %>%
  ggplot(aes(x = longitude, y = latitude)) +
  geom_tile(aes(fill = humidity_mean)) +
  theme(legend.position = "bottom") +
  scale_fill_gradient(low = "yellow", high = "red") + ggtitle("humidity_mean")

p4 <- summary_stat %>%
  ggplot(aes(x = longitude, y = latitude)) +
  geom_tile(aes(fill = humidity_variance)) +
  theme(legend.position = "bottom") +
  scale_fill_gradient(low = "yellow", high = "red") +␣
 ↪ggtitle("humidity_variance")
```

[26]:
```
grid.arrange(p1, p2, p3, p4, ncol = 2)
```



Possiamo anche creare una GIF con l'evoluzione di una variabile nel tempo e nello spazio. Ma fate attenzione che ci mette un po' (un paio di minuti sul mio mac)

[27]:
```
library(gganimate)
library(magick)
p <- dataset_clima %>%
   filter(time < 365*2-1)%>%  ggplot(aes(x = longitude, y = latitude, frame =␣
 ↪time)) +
  geom_tile(aes(fill = minimum_temperature))
anim <- p + transition_time(time) + theme(legend.position = "bottom") +␣
 ↪scale_fill_gradient2(low = "yellow",mid = "white", high = "red") +␣
 ↪labs(title = "Time: {frame_time}")
anim_save("animated_plot.gif", animation = anim,  fps = 2, nframes=100)
```

```
#gif <- image_read("animated_plot.gif")
#print(gif)
```

[28]:
```
gif <- image_read("animated_plot.gif")
print(gif)
```

```
# A tibble: 100 x 7
   format width height colorspace matte filesize density
   <chr>  <int>  <int>
<chr>      <lgl>
<int> <chr>
 1 GIF      480    480 sRGB       FALSE        0 72x72
 2 GIF      480    480 sRGB       FALSE        0 72x72
 3 GIF      480    480 sRGB       FALSE        0 72x72
 4 GIF      480    480 sRGB       FALSE        0 72x72
 5 GIF      480    480 sRGB       FALSE        0 72x72
 6 GIF      480    480 sRGB       FALSE        0 72x72
 7 GIF      480    480 sRGB       FALSE        0 72x72
 8 GIF      480    480 sRGB       FALSE        0 72x72
 9 GIF      480    480 sRGB       FALSE        0 72x72
10 GIF      480    480 sRGB       FALSE        0 72x72
# i 90 more rows
```

Possiamo vedere un'animazione di un paio di serie

[29]:
```
# install.packages("Ecdat")
library(Ecdat)
# install.packages("tidyverse")
library(tidyverse)
# install.packages("gganimate")
library(gganimate)
# install.packages("remotes")
# remotes::install_github("R-CoderDotCom/ggcats@main")
library(ggcats)

dat1 <- dataset_clima[seq(721, 1000000, by = nstaz), "minimum_temperature"]
dat2 <- dataset_clima[seq(824, 1000000, by = nstaz), "minimum_temperature"]
# Animation
dat <- data.frame(
  times = rep(1:length(dat1),2),
  y = c(dat1, dat2),
  staz = c(rep("S1", each = length(dat1)), rep("S2", each = length(dat1)))
)
ggplot(dat, aes(x = times, y = y, group = staz, color = staz)) +
  geom_line(size = 2) +
  #ggtitle("ggcats, a core package of the memeverse") +
#  geom_cat(aes(cat = cat), size = 5) +
  transition_reveal(times)
```

o fare la stessa cosa con dei gatti

[46]:
```
cat_name <-  c(
  "nyancat", "bongo",
  "colonel", "grumpy",
  "hipster", "lil_bub",
  "maru", "mouth",
  "pop", "pop_close",
  "pusheen", "pusheen_pc",
  "toast", "venus",
  "shironeko"
)

dat$cats <- c(rep(cat_name[1], each = length(dat1)), rep(cat_name[11], each =␣
 ↪length(dat1)))
ggplot(dat, aes(x = times, y = y, group = staz, color = staz)) +
  geom_line(size = 2) +
  # ggtitle("ggcats, a core package of the memeverse") +
  geom_cat(aes(cat = cats), size = 5) +
  transition_reveal(times)
```

[52]:
```
#grid <- expand.grid(1:5, 3:1)

#df <- data.frame(
#  x = grid[, 1],
#  y = grid[, 2],
#  image = c(
#    "nyancat", "bongo",
#    "colonel", "grumpy",
#    "hipster", "lil_bub",
#    "maru", "mouth",
#    "pop", "pop_close",
#    "pusheen", "pusheen_pc",
#    "toast", "venus",
#    "shironeko"
#  )
#)

#ggplot(df) +
#  geom_cat(aes(x, y, cat = image), size = 5) +
#  geom_text(aes(x, y - 0.5, label = image), size = 2.5) +
#  xlim(c(0.25, 5.5)) +
#  ylim(c(0.25, 3.5))
```

DOMANDE

1. Prendete un punto qualsiasi della mappa, e modellizzate una qualsiasi serie con un AR(1), con una media costante, ma diversa da zero. Eliminate gli ultimi 100 punti, e prevedeteli con il modello (distribuzione a posteriori). Valutare se i residui $(\hat{w}_i)$ sono un runore bianco.

Potete scrivere l'MCMC oppure utilizzare STAN

2. Fate la stessa cosa del punto precedente, ma mettete una media

$$E(y_t) = \mu + A\sin(2\pi\frac{t}{365} + c)$$

con A e c dei parametri da stimare.

3. Calcolate MSE per decidere quale dei modelli è migliore

Risposta 1

Decido di usare STAN. La prima cosa che dovete fare in STAN è scrivere il file .stan che contenga il modello (lo trovate in cartella come file esecitazione_6_file1_test.stan). Implemento il modello

$$(y_t - \mu) = \alpha(y_{t-1} - \mu) + w_t \quad w_t \sim N(0, \sigma^2)$$

Come test, voglio vedere se il modello è un AR(1) oppure un random walk. Questo si riduce a valutare se $|\alpha| < 1$ oppure $\alpha = 1$. Se assumo che la variabile α sia continua, non è possibile valutare

$$P(\alpha = 1|\mathbf{y}) = 0$$

Posso risulvere il problema definendo α come una variabile mista con massa di probabilità su 1. Definisco

$$\alpha = min(\alpha^*, 1)$$

con

$$\alpha^* \sim U(-1, 2)$$

In questo modo, ho che

$$P(\alpha = 1) = \int_1^2 f(\alpha^*)d\alpha^* = \frac{1}{3}$$

e sto dicendo che a-priori la probabilità che sia un random walk è del 33%. Le restanti prior sono

$$\sigma^2 \sim IG(1, 1) \qquad \mu \sim N(0, 10000)$$

Ricordate che la distribuzione della marginale di y_1 in un AR(1) è

$$y_1 \sim N(\mu, \frac{\sigma^2}{1 - \alpha^2})$$

Se $\alpha = 1$ la varianza va a infinito. Per risolvere il problema, definisco il modello trattando y_1 come valore noto.

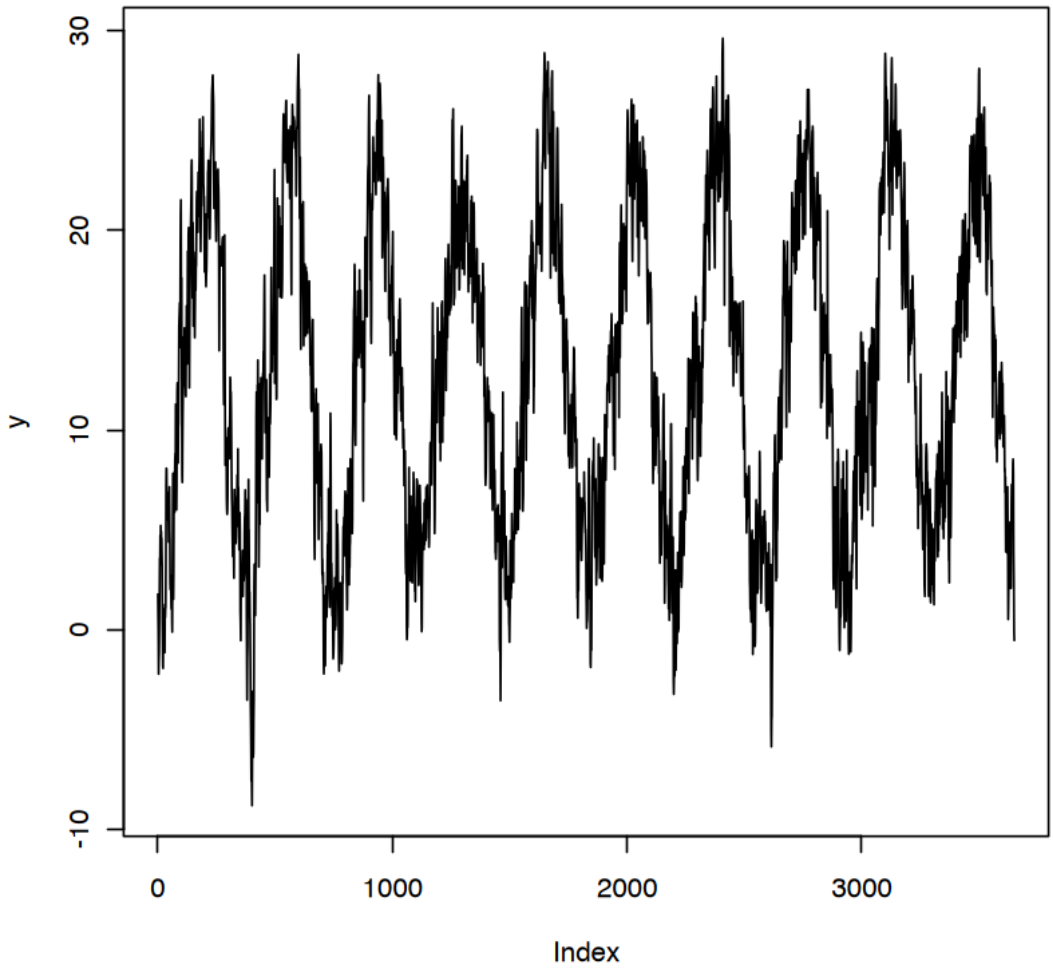
Il modello che ho implementato è mostrato qui sotto

```
data {
  int<lower=0> n_pred;            // number of point to predict
  int<lower=0> T;              // number of time points
  real y[T];                   // observed data
}

parameters {
  real mu;                    // AR(1) mean
```

```
  real alpha;                  // AR(1) coefficient
  real<lower=0> sigma2;       // standard deviation of the noise
}

transformed parameters {
  real<lower=0> sigma;       // standard deviation of the noise
  sigma = sqrt(sigma2);      // derive sigma as the square root of sigma^2
  real alpha_min;
  alpha_min = fmin(alpha, 1);
}

model {
  alpha ~ uniform(-1, 2);        // prior for AR(1) coefficient
  sigma2 ~ inv_gamma(1, 1);    // prior for the standard deviation
  mu ~ normal(0,10000^0.5);

  //y[1] ~ normal(mu, sigma/(1-alpha^2)^0.5);
  for (t in 2:T) {
    y[t] ~ normal(mu +   alpha_min * (y[t-1]-mu), sigma);  // AR(1) process
  }
}

generated quantities {
  real y_pred[n_pred];            // predicted values

  y_pred[1] = normal_rng(mu +   alpha_min * (y[T]-mu), sigma);
  for (t in 2:n_pred) {
    y_pred[t] = normal_rng(mu +   alpha_min * (y_pred[t-1]-mu), sigma);
  }
}
```

Prendo una stazione, che dovrebbe essere torino, e disegno la sua serie

[31]:
```
w <- seq(745, nrow(dataset_clima), by = nstaz)
y <- dataset_clima$mean_temperature[w]
plot(y, type="l")
```



compilo il codice e ottengo le catene

[32]:
```
library(rstan)
stan_model <- stan_model("esercitazione_6_file1_test.stan")
```

[33]:
```
nmiss <- 100
nobs <- length(y) - nmiss
data_model <- y[1:(length(y) - nmiss)]
data_miss <- y[-c(1:(length(y) - nmiss))]
length(data_miss)
data_list <- list(
  T = length(data_model),
  y = data_model,
```

```
  n_pred = nmiss
  )

fit <- sampling(stan_model, data = data_list, chains = 1, iter = 3000, warmup =␣
 ↪1000, seed = 123)
```

100

```
SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 0.000852 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 8.52
seconds.
Chain 1: Adjust your expectations accordingly!
Chain 1:
Chain 1:
Chain 1: Iteration:    1 / 3000 [  0%]  (Warmup)
Chain 1: Iteration:  300 / 3000 [ 10%]  (Warmup)
Chain 1: Iteration:  600 / 3000 [ 20%]  (Warmup)
Chain 1: Iteration:  900 / 3000 [ 30%]  (Warmup)
Chain 1: Iteration: 1001 / 3000 [ 33%]  (Sampling)
Chain 1: Iteration: 1300 / 3000 [ 43%]  (Sampling)
Chain 1: Iteration: 1600 / 3000 [ 53%]  (Sampling)
Chain 1: Iteration: 1900 / 3000 [ 63%]  (Sampling)
Chain 1: Iteration: 2200 / 3000 [ 73%]  (Sampling)
Chain 1: Iteration: 2500 / 3000 [ 83%]  (Sampling)
Chain 1: Iteration: 2800 / 3000 [ 93%]  (Sampling)
Chain 1: Iteration: 3000 / 3000 [100%]  (Sampling)
Chain 1:
Chain 1:  Elapsed Time: 38.583 seconds (Warm-up)
Chain 1:                5.784 seconds (Sampling)
Chain 1:                44.367 seconds (Total)
Chain 1:
```

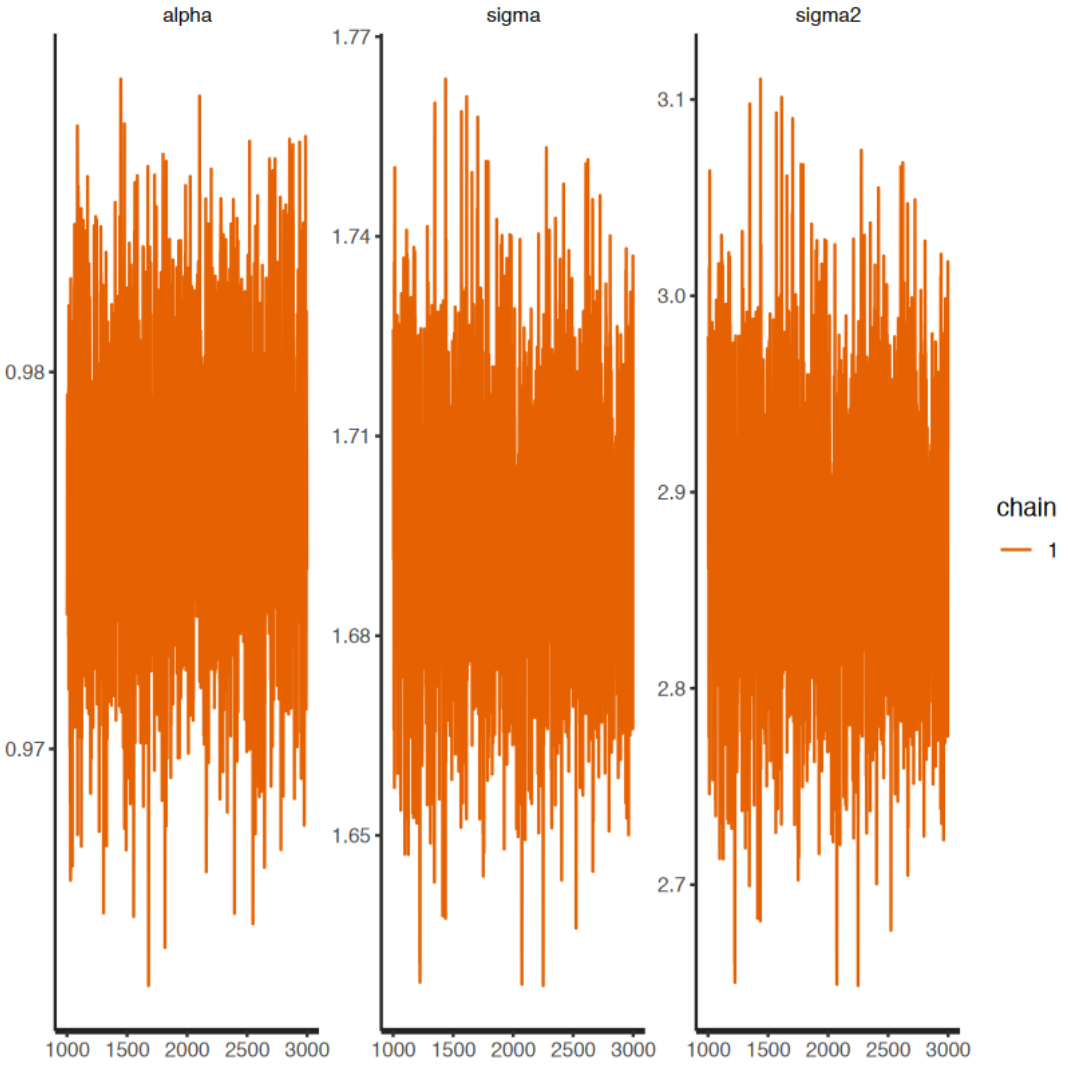
[34]:
```
# Summary of the fitted parameters
print(fit, pars = c("alpha", "sigma", "sigma2", "y_pred"))

# Plot diagnostics
traceplot(fit, pars = c("alpha", "sigma", "sigma2"))

# Extract samples
samples <- extract(fit)
```

```
Inference for Stan model: anon_model.
1 chains, each with iter=3000; warmup=1000; thin=1;
post-warmup draws per chain=2000, total post-warmup draws=2000.

              mean se_mean    sd  2.5%   25%   50%   75% 97.5% n_eff Rhat
```

alpha	0.98	0.00	0.00	0.97	0.97	0.98	0.98	0.98	1690	1
sigma	1.70	0.00	0.02	1.65	1.68	1.70	1.71	1.73	1210	1
sigma2	2.87	0.00	0.07	2.74	2.83	2.87	2.92	3.01	1212	1
y_pred[1]	19.08	0.04	1.66	15.78	17.95	19.05	20.24	22.19	1927	1
y_pred[2]	19.00	0.05	2.34	14.37	17.48	19.04	20.62	23.47	2117	1
y_pred[3]	18.86	0.07	2.83	13.25	16.93	18.86	20.82	24.41	1802	1
y_pred[4]	18.68	0.08	3.19	12.48	16.57	18.60	20.88	24.79	1745	1
y_pred[5]	18.60	0.08	3.54	11.51	16.22	18.52	21.07	25.38	1816	1
y_pred[6]	18.52	0.09	3.87	10.63	15.98	18.47	21.10	26.11	1755	1
y_pred[7]	18.42	0.09	4.06	9.81	15.80	18.45	21.08	26.51	1911	1
y_pred[8]	18.31	0.10	4.35	9.55	15.48	18.41	21.31	26.68	1934	1
y_pred[9]	18.21	0.11	4.59	9.08	15.23	18.21	21.35	27.22	1904	1
y_pred[10]	18.09	0.11	4.75	8.77	14.95	18.05	21.54	26.96	1871	1
y_pred[11]	18.00	0.11	4.90	8.01	14.73	18.04	21.49	27.01	1875	1
y_pred[12]	17.87	0.12	5.11	7.47	14.40	18.12	21.26	27.62	1683	1
y_pred[13]	17.77	0.13	5.24	7.24	14.10	18.04	21.38	27.41	1633	1
y_pred[14]	17.64	0.12	5.44	7.00	13.86	17.82	21.22	28.38	1901	1
y_pred[15]	17.53	0.13	5.60	6.36	13.68	17.70	21.32	28.41	1960	1
y_pred[16]	17.39	0.13	5.71	5.91	13.48	17.43	21.36	28.59	1979	1
y_pred[17]	17.31	0.13	5.85	5.93	13.25	17.27	21.37	28.65	2021	1
y_pred[18]	17.24	0.13	5.97	5.75	13.12	17.40	21.44	28.67	2006	1
y_pred[19]	17.09	0.14	6.11	5.04	12.99	17.12	21.29	28.99	2030	1
y_pred[20]	17.01	0.14	6.20	4.90	12.78	17.25	21.25	29.14	2063	1
y_pred[21]	16.89	0.14	6.31	4.69	12.61	16.94	21.31	29.17	2001	1
y_pred[22]	16.80	0.14	6.36	4.78	12.46	16.87	21.17	29.50	1939	1
y_pred[23]	16.75	0.15	6.47	4.15	12.23	16.82	21.21	29.59	1888	1
y_pred[24]	16.70	0.15	6.55	4.17	12.01	16.77	21.20	30.05	1844	1
y_pred[25]	16.62	0.16	6.67	3.76	11.99	16.74	21.21	30.16	1844	1
y_pred[26]	16.53	0.16	6.73	4.03	11.70	16.51	21.09	29.85	1843	1
y_pred[27]	16.49	0.16	6.81	3.63	11.84	16.44	21.15	29.79	1810	1
y_pred[28]	16.37	0.16	6.94	2.64	11.69	16.22	21.06	29.76	1786	1
y_pred[29]	16.28	0.17	7.02	2.56	11.63	16.15	21.06	29.71	1747	1
y_pred[30]	16.20	0.17	7.12	2.54	11.13	16.07	21.06	30.10	1806	1
y_pred[31]	16.14	0.17	7.26	2.54	10.85	16.17	21.16	30.38	1843	1
y_pred[32]	16.12	0.17	7.23	2.71	11.18	16.02	20.96	29.98	1815	1
y_pred[33]	16.02	0.17	7.23	2.40	11.00	15.94	20.93	29.84	1827	1
y_pred[34]	15.92	0.17	7.33	2.21	10.89	15.82	20.75	30.45	1819	1
y_pred[35]	15.80	0.17	7.33	2.23	10.71	15.67	20.63	30.18	1869	1
y_pred[36]	15.74	0.17	7.39	2.21	10.60	15.72	20.59	30.36	1851	1
y_pred[37]	15.70	0.17	7.38	1.87	10.68	15.58	20.59	29.98	1866	1
y_pred[38]	15.59	0.17	7.41	1.86	10.50	15.29	20.57	30.08	1903	1
y_pred[39]	15.52	0.17	7.37	1.71	10.56	15.36	20.45	30.23	1947	1
y_pred[40]	15.46	0.17	7.44	1.53	10.52	15.36	20.42	30.24	1972	1
y_pred[41]	15.39	0.17	7.44	0.99	10.37	15.24	20.40	30.27	1982	1
y_pred[42]	15.34	0.17	7.48	0.95	10.36	15.22	20.24	30.38	2003	1
y_pred[43]	15.28	0.17	7.60	0.30	9.99	15.23	20.21	30.58	2021	1
y_pred[44]	15.28	0.17	7.61	0.41	10.21	15.18	20.35	30.47	2053	1
y_pred[45]	15.21	0.17	7.73	0.28	9.96	15.02	20.33	30.34	2091	1

y_pred[46]	15.17	0.17	7.71	0.03	10.04	14.93	20.34	30.53	2098	1
y_pred[47]	15.10	0.17	7.74	-0.08	9.87	14.93	20.18	30.83	2067	1
y_pred[48]	15.07	0.17	7.74	-0.34	9.95	14.85	20.15	30.61	2075	1
y_pred[49]	15.05	0.17	7.75	0.00	9.86	14.93	20.20	30.26	2059	1
y_pred[50]	14.99	0.17	7.75	-0.22	9.91	14.71	20.40	30.19	2081	1
y_pred[51]	14.96	0.17	7.79	0.03	9.94	14.74	20.27	30.59	2083	1
y_pred[52]	14.91	0.17	7.76	-0.38	9.96	14.60	20.14	29.91	2105	1
y_pred[53]	14.90	0.17	7.72	0.12	9.64	14.66	19.98	29.89	2068	1
y_pred[54]	14.86	0.17	7.74	0.14	9.59	14.73	19.89	29.92	2074	1
y_pred[55]	14.82	0.17	7.73	-0.02	9.42	14.79	19.87	30.30	2090	1
y_pred[56]	14.78	0.17	7.77	-0.24	9.76	14.74	19.78	30.19	2128	1
y_pred[57]	14.73	0.17	7.72	-0.35	9.76	14.81	19.69	30.16	2117	1
y_pred[58]	14.66	0.17	7.76	-0.28	9.52	14.61	19.61	30.37	2068	1
y_pred[59]	14.68	0.17	7.75	0.00	9.55	14.72	19.71	30.13	2030	1
y_pred[60]	14.66	0.17	7.77	-0.46	9.54	14.68	19.61	30.33	2044	1
y_pred[61]	14.53	0.17	7.75	-0.21	9.24	14.42	19.70	29.91	2009	1
y_pred[62]	14.56	0.17	7.77	-0.32	9.46	14.49	19.66	30.13	2005	1
y_pred[63]	14.55	0.17	7.81	-0.09	9.25	14.43	19.50	30.66	2029	1
y_pred[64]	14.59	0.17	7.75	-0.06	9.26	14.38	19.43	30.61	2013	1
y_pred[65]	14.60	0.17	7.78	-0.14	9.08	14.34	19.66	30.88	2021	1
y_pred[66]	14.52	0.17	7.78	-0.30	9.35	14.32	19.50	30.43	2030	1
y_pred[67]	14.48	0.17	7.83	-0.49	9.27	14.34	19.48	30.78	2066	1
y_pred[68]	14.43	0.17	7.82	-0.90	9.10	14.34	19.42	30.56	2045	1
y_pred[69]	14.38	0.17	7.82	-0.73	9.19	14.32	19.38	30.44	2031	1
y_pred[70]	14.30	0.17	7.81	-0.83	9.23	14.30	19.46	30.16	2008	1
y_pred[71]	14.26	0.18	7.91	-1.40	9.00	14.43	19.41	30.21	2041	1
y_pred[72]	14.16	0.18	7.94	-1.47	8.88	14.31	19.23	30.03	2048	1
y_pred[73]	14.13	0.17	7.91	-1.34	8.81	14.27	19.19	30.52	2065	1
y_pred[74]	14.10	0.17	7.89	-1.50	8.84	14.06	19.09	30.42	2069	1
y_pred[75]	14.05	0.17	7.92	-1.48	8.75	14.03	19.20	30.20	2080	1
y_pred[76]	14.03	0.17	7.92	-1.13	8.71	14.05	19.08	29.90	2088	1
y_pred[77]	14.01	0.17	7.91	-1.22	8.68	14.15	19.19	30.37	2095	1
y_pred[78]	13.97	0.18	7.98	-1.22	8.53	13.92	19.29	29.75	2043	1
y_pred[79]	13.93	0.18	7.95	-1.12	8.43	13.83	19.31	29.43	2020	1
y_pred[80]	13.91	0.18	7.89	-1.30	8.35	13.86	19.17	29.32	1990	1
y_pred[81]	13.92	0.18	7.90	-1.24	8.30	13.88	19.07	29.54	1961	1
y_pred[82]	13.94	0.18	7.88	-1.22	8.34	13.97	19.23	29.76	1942	1
y_pred[83]	13.87	0.18	7.87	-1.63	8.37	13.81	19.07	29.31	1980	1
y_pred[84]	13.95	0.18	7.87	-1.16	8.48	13.81	19.22	29.51	1978	1
y_pred[85]	13.95	0.18	7.87	-1.52	8.66	13.71	19.23	29.69	2017	1
y_pred[86]	14.02	0.18	7.88	-1.08	8.53	13.94	19.06	29.70	1968	1
y_pred[87]	14.04	0.17	7.86	-1.29	8.71	13.78	19.23	29.90	2020	1
y_pred[88]	13.98	0.18	7.89	-0.96	8.39	13.73	19.36	30.00	1981	1
y_pred[89]	13.94	0.18	7.91	-0.88	8.33	13.85	19.23	29.34	2009	1
y_pred[90]	13.90	0.18	7.90	-1.18	8.35	13.67	19.27	29.47	2014	1
y_pred[91]	13.83	0.18	7.96	-1.28	8.39	13.71	19.02	29.69	2041	1
y_pred[92]	13.75	0.17	7.99	-1.64	8.57	13.61	19.35	29.46	2089	1
y_pred[93]	13.68	0.18	7.98	-1.75	8.38	13.56	19.22	29.09	2038	1

```
y_pred[94]  13.62    0.18 8.08 -1.91  8.27 13.38 19.13 29.25  2052    1
y_pred[95]  13.64    0.18 8.03 -1.68  8.36 13.59 19.07 28.91  2007    1
y_pred[96]  13.61    0.18 8.06 -2.19  8.36 13.57 19.15 29.30  1999    1
y_pred[97]  13.56    0.18 8.11 -2.44  8.08 13.57 19.05 29.56  1993    1
y_pred[98]  13.53    0.18 8.11 -2.38  8.20 13.58 18.97 29.62  2008    1
y_pred[99]  13.60    0.18 8.10 -2.39  8.29 13.91 19.09 29.06  2002    1
y_pred[100] 13.60    0.18 8.10 -2.22  8.30 13.78 19.09 29.18  2011    1

Samples were drawn using NUTS(diag_e) at Thu Nov 14 16:02:48 2024.
For each parameter, n_eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor on split chains (at
convergence, Rhat=1).
```



Sia dagli intervalli di credibilità di α che le catene (che non vanno mai sopra 1), potete vedere che

avete

$$P(random - walk|\mathbf{y}) \approx 0$$

Vediamo i plot delle previsioni e confrontiamoli con i valori veri

[35]:
```
str(samples)
samp_pred = samples$y_pred
```

```
List of 7
 $ mu       : num [1:2000(1d)] 13.9 14.6 11.6 11.1 14.4 …
  ..- attr(*, "dimnames")=List of 1
  .. ..$ iterations: NULL
 $ alpha    : num [1:2000(1d)] 0.977 0.981 0.978 0.973 0.972 …
  ..- attr(*, "dimnames")=List of 1
  .. ..$ iterations: NULL
 $ sigma2   : num [1:2000(1d)] 2.79 3 2.91 2.83 2.88 …
  ..- attr(*, "dimnames")=List of 1
  .. ..$ iterations: NULL
 $ sigma    : num [1:2000(1d)] 1.67 1.73 1.71 1.68 1.7 …
  ..- attr(*, "dimnames")=List of 1
  .. ..$ iterations: NULL
 $ alpha_min: num [1:2000(1d)] 0.977 0.981 0.978 0.973 0.972 …
  ..- attr(*, "dimnames")=List of 1
  .. ..$ iterations: NULL
 $ y_pred   : num [1:2000, 1:100] 16.9 20.9 18 18.9 20.2 …
  ..- attr(*, "dimnames")=List of 2
  .. ..$ iterations: NULL
  .. ..$           : NULL
 $ lp__     : num [1:2000(1d)] -3649 -3651 -3649 -3650 -3649 …
  ..- attr(*, "dimnames")=List of 1
  .. ..$ iterations: NULL
```

[36]:
```
mean_pred <- c(apply(samp_pred, 2, mean))
q1 <- apply(samp_pred, 2,function(x) quantile(x, probs = 0.025))
q2 <- apply(samp_pred, 2, function(x) quantile(x, probs = 1-0.025))

data_plot_tot <- data.frame(
  y = y,
  time = 1:length(y),
  obs_miss = c(rep("Obs",  nobs), rep("Miss",  nmiss)),
  mean = c(rep(NA, nobs), mean_pred),
  q1 = c(rep(NA, nobs), q1),
  q2 = c(rep(NA, nobs), q2)
  )

data_plot_tot %>%
  slice((length(y) -400): length(y))%>%
  ggplot(aes(x = time, y = y, col= obs_miss)) +
```

```
  geom_line() +
  geom_line(aes(y = mean), col="red")+
  geom_hline(yintercept = mean(samples$mu), lwd = 1, col=1) +
  geom_ribbon(aes(ymin = q1, ymax = q2), alpha = 0.2)
```

```
Warning message:
"Removed 301 rows containing missing values or values outside the scale
range
(`geom_line()`)."
Warning message in max(ids, na.rm = TRUE):
"nessun argomento non-mancante al massimo; si restituisce -Inf"
```

Le linea rossa della media delle previsione (la media condizionata) tende a μ poichè la dsitributione

condizionata è

$$(y_{n+h} - \mu)|\mathbf{y} \sim N\left(\alpha^h(y_n - \mu), \sigma^2 \sum_{j=1}^{h} \alpha^{2j}\right)$$

e quindi

$$y_{n+h}|\mathbf{y} \sim N\left(\mu + \alpha^h(y_n - \mu), \sigma^2 \sum_{j=1}^{h} \alpha^{2j}\right)$$

dove x_n è l'ultimo punto del vettore delle variabili osservate $\mathbf{x}$

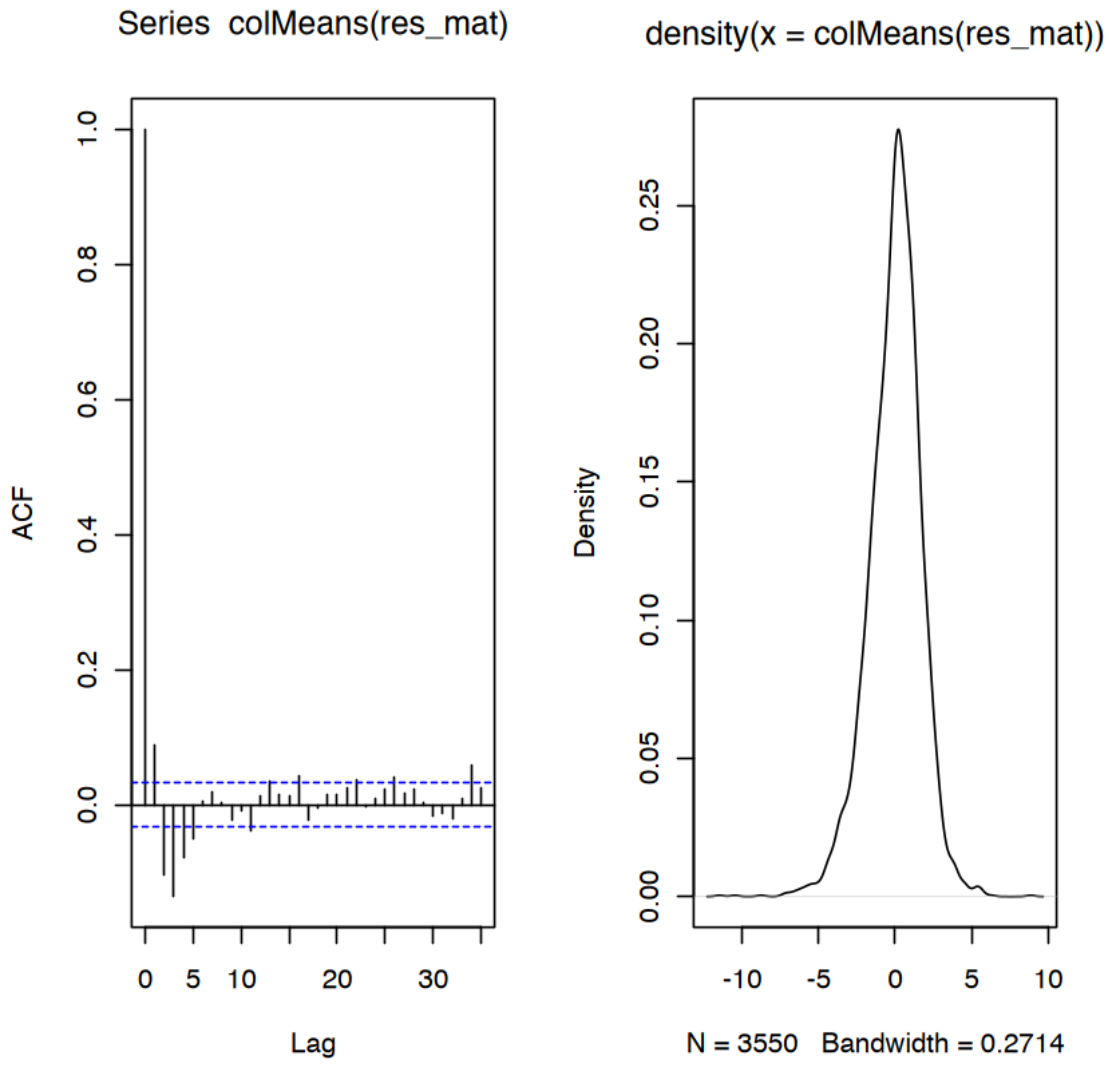
Calcoliamo anche i residui (campioni), che sono

$$\hat{w}_t^b = y_t - \mu^b - \alpha^b(y_{t-1} - \mu^b)$$

[37]:
```
res_mat <- matrix(NA, ncol = nobs, nrow = nrow(samples$alpha))

for(isim in 1:nrow(samples$alpha))
{

  res_mat[isim, 1] <- y[1] - samples$mu[isim]
  for(i in 2:nobs)
  {
    res_mat[isim, i] <- y[i] - samples$mu[isim] - samples$alpha[isim] * (y[i-1]␣
 ↪- samples$mu[isim])
  }

}
```

calcolo la la media e valutiamo il correlogramma e la stima della distribuzione

[38]:
```
par(mfrow=c(1,2))
acf(colMeans(res_mat))
plot(density(colMeans(res_mat)))
par(mfrow = c(1, 1))
```



Risposta2

In questo modello, facciamo le stesse cose, ma modificando il codice. Le prior le trovate nel codice

```
data {
  int<lower=0> n_pred;            // number of point to predict
  int<lower=0> T;              // number of time points
  real y[T];                   // observed data
}

parameters {
  real mu;                    // AR(1) mean
  real alpha;                    // AR(1) coefficient
  real<lower=0> sigma2;        // standard deviation of the noise
```

```
  real<lower=0> A;
  real<lower=0, upper= 2*pi()> c;

}

transformed parameters {
  real<lower=0> sigma;       // standard deviation of the noise
  sigma = sqrt(sigma2);      // derive sigma as the square root of sigma^2
}

model {
  alpha ~ uniform(-1, 1);        // prior for AR(1) coefficient
  sigma2 ~ inv_gamma(1, 1);    // prior for the standard deviation
  mu ~ normal(0,10000^0.5);
  A ~ uniform(0,60);
  c ~ uniform(0,2.0 * pi());

  y[1] ~ normal(mu + A * sin(2.0 * pi()*1/365 +c  ), sigma/(1-alpha^2)^0.5);
  for (t in 2:T) {
    y[t] ~ normal(mu + A * sin(2*pi()*t/365 +c  ) + alpha * (y[t-1]-mu - A*sin(2*pi()*(t-1)/365
  }
}

generated quantities {
  real y_pred[n_pred];            // predicted values

  y_pred[1] = normal_rng(mu + A*sin(2*pi()*(T+1)/365 +c  ) + alpha * (y[T]-mu - A*sin(2*pi()*T/
  for (t in 2:n_pred) {
    y_pred[t] = normal_rng(mu + A*sin(2*pi()*(T+t)/365 +c  )  + alpha * (y_pred[t-1]-mu - A*sin
  }
}
```

[39]:
```
stan_model_2 <- stan_model("esercitazione_6_file2.stan")
fit_2 <- sampling(stan_model_2, data = data_list, chains = 1, iter = 3000,␣
 ↪warmup = 1000, seed = 123)
```

```
SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 0.001473 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take
14.73 seconds.
Chain 1: Adjust your expectations accordingly!
Chain 1:
Chain 1:
Chain 1: Iteration:    1 / 3000 [  0%]  (Warmup)
Chain 1: Iteration:  300 / 3000 [ 10%]  (Warmup)
Chain 1: Iteration:  600 / 3000 [ 20%]  (Warmup)
```

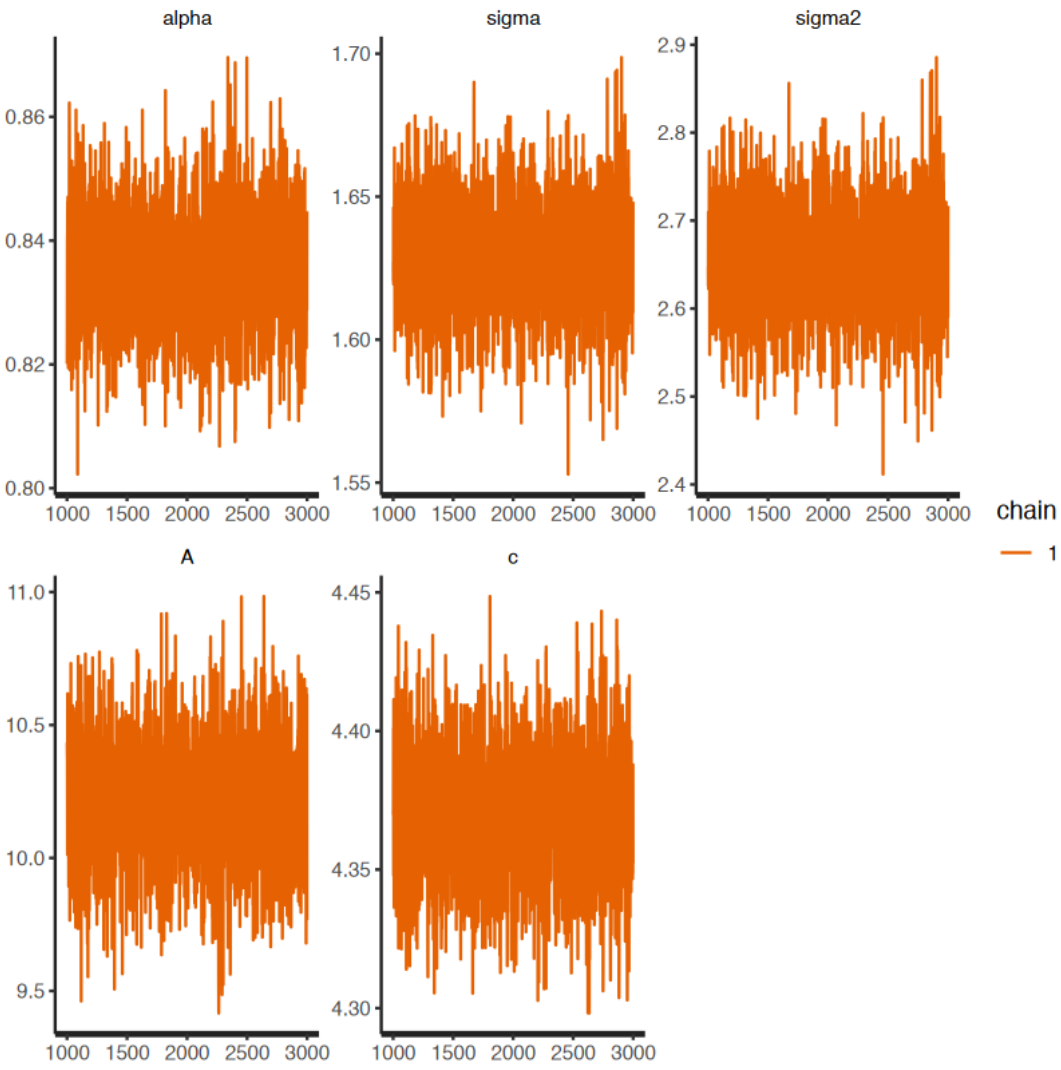
```
Chain 1: Iteration:  900 / 3000 [ 30%]  (Warmup)
Chain 1: Iteration: 1001 / 3000 [ 33%]  (Sampling)
Chain 1: Iteration: 1300 / 3000 [ 43%]  (Sampling)
Chain 1: Iteration: 1600 / 3000 [ 53%]  (Sampling)
Chain 1: Iteration: 1900 / 3000 [ 63%]  (Sampling)
Chain 1: Iteration: 2200 / 3000 [ 73%]  (Sampling)
Chain 1: Iteration: 2500 / 3000 [ 83%]  (Sampling)
Chain 1: Iteration: 2800 / 3000 [ 93%]  (Sampling)
Chain 1: Iteration: 3000 / 3000 [100%]  (Sampling)
Chain 1:
Chain 1:  Elapsed Time: 18.787 seconds (Warm-up)
Chain 1:                8.169 seconds (Sampling)
Chain 1:                26.956 seconds (Total)
Chain 1:
```

[40]:
```
# Summary of the fitted parameters
print(fit_2, pars = c("alpha", "sigma", "sigma2", "A", "c","y_pred"))

# Plot diagnostics
traceplot(fit_2, pars = c("alpha", "sigma", "sigma2", "A", "c"))

# Extract samples
samples_2 <- extract(fit_2)
samp_pred_2 <- samples_2$y_pred
```

```
Inference for Stan model: anon_model.
1 chains, each with iter=3000; warmup=1000; thin=1;
post-warmup draws per chain=2000, total post-warmup draws=2000.

             mean se_mean   sd  2.5%   25%   50%   75% 97.5% n_eff Rhat
alpha        0.84    0.00 0.01  0.82  0.83  0.84  0.84  0.85  2377    1
sigma        1.63    0.00 0.02  1.59  1.62  1.63  1.64  1.67  2545    1
sigma2       2.65    0.00 0.06  2.53  2.61  2.65  2.70  2.78  2542    1
A           10.20    0.01 0.23  9.75 10.04 10.20 10.36 10.66  2088    1
c            4.37    0.00 0.02  4.32  4.35  4.37  4.38  4.42  2398    1
y_pred[1]   18.85    0.04 1.65 15.67 17.71 18.84 19.97 22.08  1833    1
y_pred[2]   18.52    0.05 2.11 14.45 17.04 18.55 19.95 22.57  2012    1
y_pred[3]   18.22    0.05 2.37 13.47 16.65 18.20 19.82 22.74  1931    1
y_pred[4]   17.92    0.06 2.57 12.79 16.18 17.95 19.59 23.01  2059    1
y_pred[5]   17.61    0.06 2.69 12.31 15.78 17.53 19.51 23.05  2149    1
y_pred[6]   17.33    0.06 2.77 12.13 15.47 17.33 19.18 22.58  2222    1
y_pred[7]   17.08    0.06 2.82 11.72 15.07 17.12 19.07 22.55  2176    1
y_pred[8]   16.86    0.06 2.92 11.20 14.87 16.83 18.82 22.63  2081    1
y_pred[9]   16.71    0.06 2.93 11.04 14.69 16.73 18.79 22.31  2079    1
y_pred[10]  16.47    0.07 2.95 10.73 14.50 16.45 18.55 22.39  2037    1
y_pred[11]  16.26    0.07 2.95 10.52 14.30 16.25 18.29 22.02  2015    1
y_pred[12]  16.07    0.07 3.01 10.35 13.99 16.07 18.17 22.06  1962    1
y_pred[13]  15.84    0.07 3.02 10.01 13.74 15.88 17.86 21.82  1960    1
```

y_pred[14]	15.68	0.07	3.01	9.98	13.64	15.65	17.70	21.88	1989	1
y_pred[15]	15.44	0.07	3.02	9.61	13.44	15.44	17.38	21.61	1921	1
y_pred[16]	15.25	0.07	3.01	9.17	13.32	15.22	17.31	21.28	1929	1
y_pred[17]	15.06	0.07	2.99	8.93	13.01	15.09	17.10	20.80	1971	1
y_pred[18]	14.92	0.07	2.97	8.80	12.92	14.96	17.02	20.50	1943	1
y_pred[19]	14.71	0.07	2.99	8.63	12.65	14.78	16.75	20.34	2072	1
y_pred[20]	14.53	0.07	2.91	8.76	12.53	14.57	16.55	20.14	1894	1
y_pred[21]	14.36	0.07	2.94	8.38	12.41	14.37	16.39	20.04	1869	1
y_pred[22]	14.22	0.07	2.96	8.45	12.18	14.20	16.20	20.25	1701	1
y_pred[23]	13.99	0.07	2.97	8.27	11.93	13.90	16.02	19.90	1772	1
y_pred[24]	13.80	0.07	2.96	8.34	11.78	13.73	15.80	19.76	1862	1
y_pred[25]	13.52	0.07	2.99	7.86	11.48	13.48	15.42	19.57	2112	1
y_pred[26]	13.36	0.07	2.96	7.67	11.33	13.37	15.33	19.20	2063	1
y_pred[27]	13.20	0.07	2.97	7.59	11.20	13.15	15.24	19.15	2056	1
y_pred[28]	13.01	0.07	2.96	7.32	10.98	13.00	14.93	19.01	2027	1
y_pred[29]	12.82	0.06	2.95	7.13	10.81	12.81	14.86	18.60	2124	1
y_pred[30]	12.70	0.06	2.97	6.91	10.66	12.67	14.66	18.58	2152	1
y_pred[31]	12.54	0.06	2.98	7.01	10.46	12.48	14.56	18.38	2108	1
y_pred[32]	12.36	0.06	3.00	6.48	10.30	12.39	14.43	18.13	2193	1
y_pred[33]	12.15	0.06	2.98	6.32	10.11	12.17	14.22	17.93	2241	1
y_pred[34]	11.98	0.06	2.97	6.06	10.07	11.95	13.98	17.70	2256	1
y_pred[35]	11.87	0.06	2.93	6.41	9.94	11.90	13.83	17.65	2175	1
y_pred[36]	11.72	0.06	2.92	6.11	9.78	11.61	13.75	17.31	2157	1
y_pred[37]	11.48	0.06	2.91	6.04	9.53	11.46	13.43	17.22	2045	1
y_pred[38]	11.36	0.06	2.91	5.51	9.43	11.41	13.32	17.29	2046	1
y_pred[39]	11.26	0.07	2.97	5.70	9.30	11.25	13.17	17.10	1988	1
y_pred[40]	11.06	0.07	2.97	5.01	9.04	11.05	12.93	17.09	1917	1
y_pred[41]	10.82	0.07	3.00	4.72	8.81	10.84	12.85	16.73	1822	1
y_pred[42]	10.61	0.07	2.92	4.95	8.60	10.63	12.62	16.33	1696	1
y_pred[43]	10.51	0.07	2.91	4.66	8.61	10.51	12.48	16.15	1993	1
y_pred[44]	10.32	0.07	2.95	4.27	8.42	10.32	12.29	15.99	1964	1
y_pred[45]	10.12	0.07	2.97	4.24	8.13	10.08	12.09	15.91	2007	1
y_pred[46]	9.95	0.07	2.97	4.28	8.00	10.04	11.97	15.72	2081	1
y_pred[47]	9.65	0.06	2.96	3.84	7.63	9.70	11.59	15.42	2085	1
y_pred[48]	9.49	0.06	2.93	3.70	7.47	9.49	11.52	15.22	2123	1
y_pred[49]	9.38	0.06	2.90	3.86	7.39	9.40	11.42	14.97	2138	1
y_pred[50]	9.24	0.06	2.95	3.49	7.25	9.26	11.31	14.71	2165	1
y_pred[51]	9.08	0.07	2.95	3.19	7.00	9.16	11.14	14.68	2050	1
y_pred[52]	8.97	0.06	2.90	3.46	6.94	8.88	10.95	14.63	2008	1
y_pred[53]	8.79	0.07	2.95	3.06	6.91	8.76	10.73	14.45	2005	1
y_pred[54]	8.67	0.07	2.96	2.80	6.67	8.71	10.68	14.54	1841	1
y_pred[55]	8.56	0.07	2.94	3.04	6.54	8.48	10.55	14.13	1788	1
y_pred[56]	8.32	0.07	2.97	2.67	6.32	8.24	10.34	14.24	1923	1
y_pred[57]	8.26	0.07	2.98	2.32	6.22	8.27	10.27	14.05	1865	1
y_pred[58]	8.10	0.07	3.03	2.32	6.03	8.14	10.16	13.94	1801	1
y_pred[59]	7.96	0.07	2.96	2.20	6.10	7.97	9.86	13.75	1839	1
y_pred[60]	7.78	0.07	2.91	2.09	5.82	7.77	9.74	13.58	1855	1
y_pred[61]	7.62	0.07	2.94	1.85	5.63	7.56	9.60	13.44	1940	1

```
y_pred[62]   7.45    0.07 2.99  1.65  5.40  7.40  9.44 13.33  1909    1
y_pred[63]   7.30    0.07 2.93  1.77  5.30  7.30  9.33 13.01  1991    1
y_pred[64]   7.20    0.07 2.99  1.24  5.20  7.17  9.24 13.05  2049    1
y_pred[65]   7.00    0.07 2.96  1.38  5.01  6.95  8.98 12.94  1743    1
y_pred[66]   6.80    0.07 3.00  1.11  4.72  6.74  8.82 12.86  1916    1
y_pred[67]   6.68    0.07 3.04  0.92  4.54  6.56  8.68 12.80  1732    1
y_pred[68]   6.60    0.07 3.06  0.51  4.49  6.55  8.63 12.55  1751    1
y_pred[69]   6.41    0.07 3.00  0.57  4.37  6.39  8.38 12.38  2103    1
y_pred[70]   6.22    0.06 3.03  0.38  4.17  6.23  8.28 12.24  2199    1
y_pred[71]   6.18    0.06 2.98  0.26  4.18  6.27  8.13 11.78  2263    1
y_pred[72]   6.03    0.06 2.94  0.22  4.01  6.10  7.99 11.84  2163    1
y_pred[73]   5.90    0.06 2.97  0.05  3.98  5.98  7.87 11.72  2191    1
y_pred[74]   5.70    0.06 2.94  0.05  3.71  5.75  7.75 11.39  2204    1
y_pred[75]   5.61    0.07 2.94 -0.12  3.64  5.71  7.60 11.43  2005    1
y_pred[76]   5.52    0.07 2.89 -0.32  3.60  5.55  7.36 11.17  1832    1
y_pred[77]   5.42    0.06 2.84 -0.08  3.50  5.42  7.29 10.92  1953    1
y_pred[78]   5.26    0.07 2.86 -0.29  3.30  5.25  7.21 10.85  1923    1
y_pred[79]   5.13    0.06 2.87 -0.39  3.21  5.16  7.13 10.74  1984    1
y_pred[80]   5.05    0.07 2.94 -0.73  3.05  5.06  7.04 10.64  1978    1
y_pred[81]   4.90    0.07 2.95 -0.83  2.97  4.94  6.89 10.48  1993    1
y_pred[82]   4.82    0.07 3.00 -1.11  2.80  4.83  6.86 10.70  1903    1
y_pred[83]   4.76    0.07 2.99 -1.22  2.77  4.81  6.84 10.42  1899    1
y_pred[84]   4.66    0.06 2.98 -1.14  2.71  4.64  6.72 10.53  2134    1
y_pred[85]   4.56    0.07 3.00 -1.38  2.48  4.56  6.61 10.41  2035    1
y_pred[86]   4.48    0.07 3.00 -1.39  2.50  4.51  6.43 10.46  2035    1
y_pred[87]   4.41    0.07 3.02 -1.66  2.42  4.36  6.45 10.38  1936    1
y_pred[88]   4.31    0.07 3.02 -1.70  2.30  4.31  6.29 10.49  1970    1
y_pred[89]   4.21    0.07 2.95 -1.72  2.27  4.19  6.13 10.05  1950    1
y_pred[90]   4.06    0.07 2.94 -1.83  2.10  4.04  5.93  9.74  1995    1
y_pred[91]   3.96    0.07 2.96 -1.68  2.01  3.93  5.92  9.86  1999    1
y_pred[92]   3.94    0.07 2.95 -1.53  1.91  3.96  5.96  9.72  1927    1
y_pred[93]   3.88    0.07 2.96 -1.95  1.83  3.86  5.87  9.62  1880    1
y_pred[94]   3.69    0.07 2.96 -1.94  1.74  3.67  5.68  9.60  1956    1
y_pred[95]   3.65    0.07 2.96 -2.10  1.64  3.61  5.63  9.60  1835    1
y_pred[96]   3.57    0.07 2.97 -2.01  1.54  3.49  5.61  9.54  1824    1
y_pred[97]   3.52    0.07 2.94 -2.08  1.48  3.56  5.52  9.33  1848    1
y_pred[98]   3.43    0.07 2.96 -2.42  1.49  3.36  5.47  9.33  1819    1
y_pred[99]   3.37    0.07 2.95 -2.28  1.31  3.36  5.36  8.86  1779    1
y_pred[100]  3.30    0.07 2.96 -2.20  1.26  3.25  5.27  9.36  1849    1

Samples were drawn using NUTS(diag_e) at Thu Nov 14 16:04:23 2024.
For each parameter, n_eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor on split chains (at
convergence, Rhat=1).
```

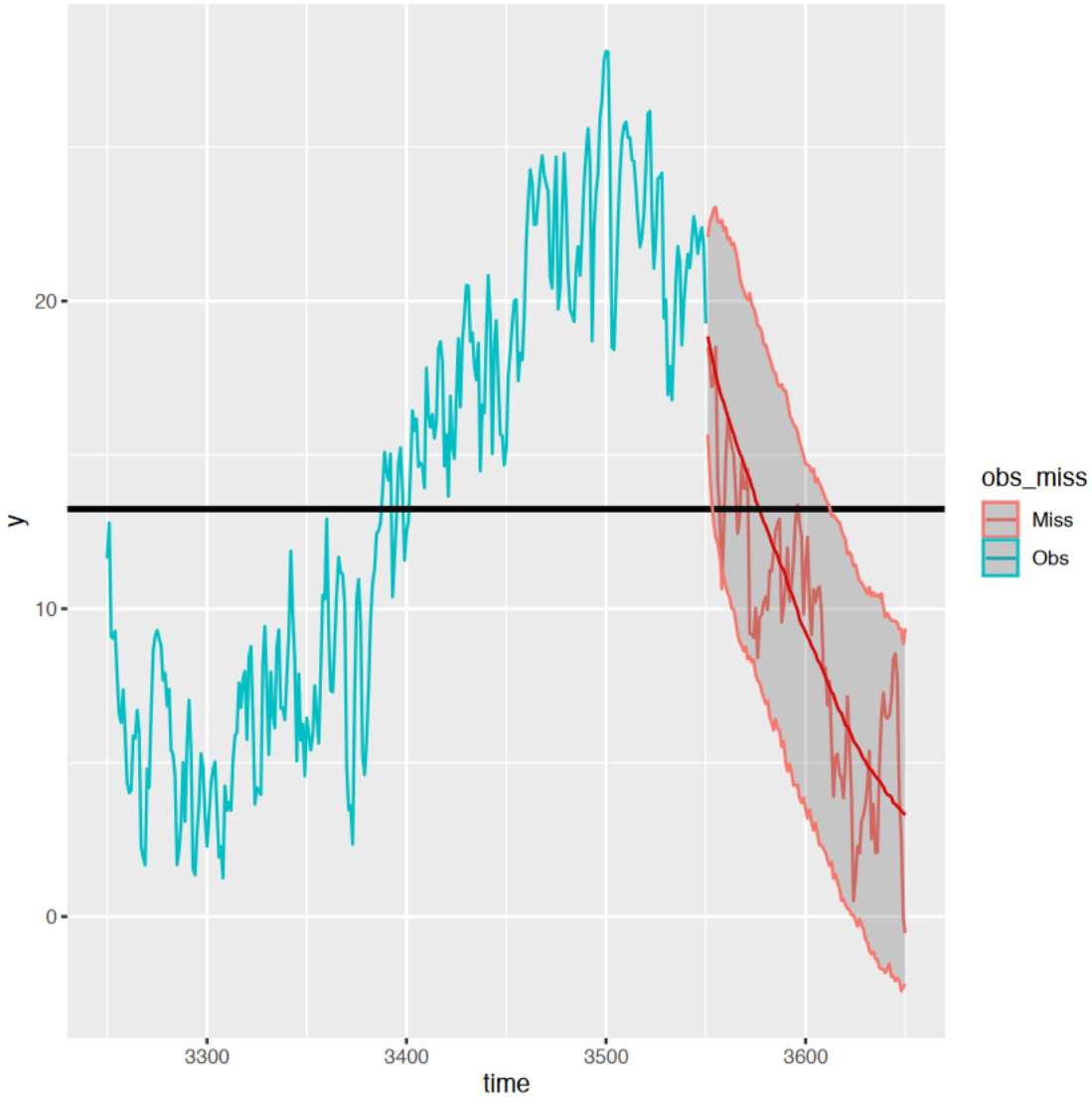


Anche in questo caso vediamo la previsione

[41]:
```
mean_pred_2 <- c(apply(samp_pred_2, 2, mean))
q1_2 <- apply(samp_pred_2, 2, function(x) quantile(x, probs = 0.025))
q2_2 <- apply(samp_pred_2, 2, function(x) quantile(x, probs = 1 - 0.025))

data_plot_tot_2 <- data.frame(
  y = y,
  time = 1:length(y),
  obs_miss = c(rep("Obs", nobs), rep("Miss", nmiss)),
  mean = c(rep(NA, nobs), mean_pred_2),
  q1 = c(rep(NA, nobs), q1_2),
  q2 = c(rep(NA, nobs), q2_2)
```

```
)

data_plot_tot_2 %>%
  slice((length(y) - 400):length(y)) %>%
  ggplot(aes(x = time, y = y, col = obs_miss)) +
  geom_line() +
  geom_line(aes(y = mean), col = "red") +
  geom_hline(yintercept = mean(samples$mu), lwd = 1, col = 1) +
  geom_ribbon(aes(ymin = q1, ymax = q2), alpha = 0.2)
```

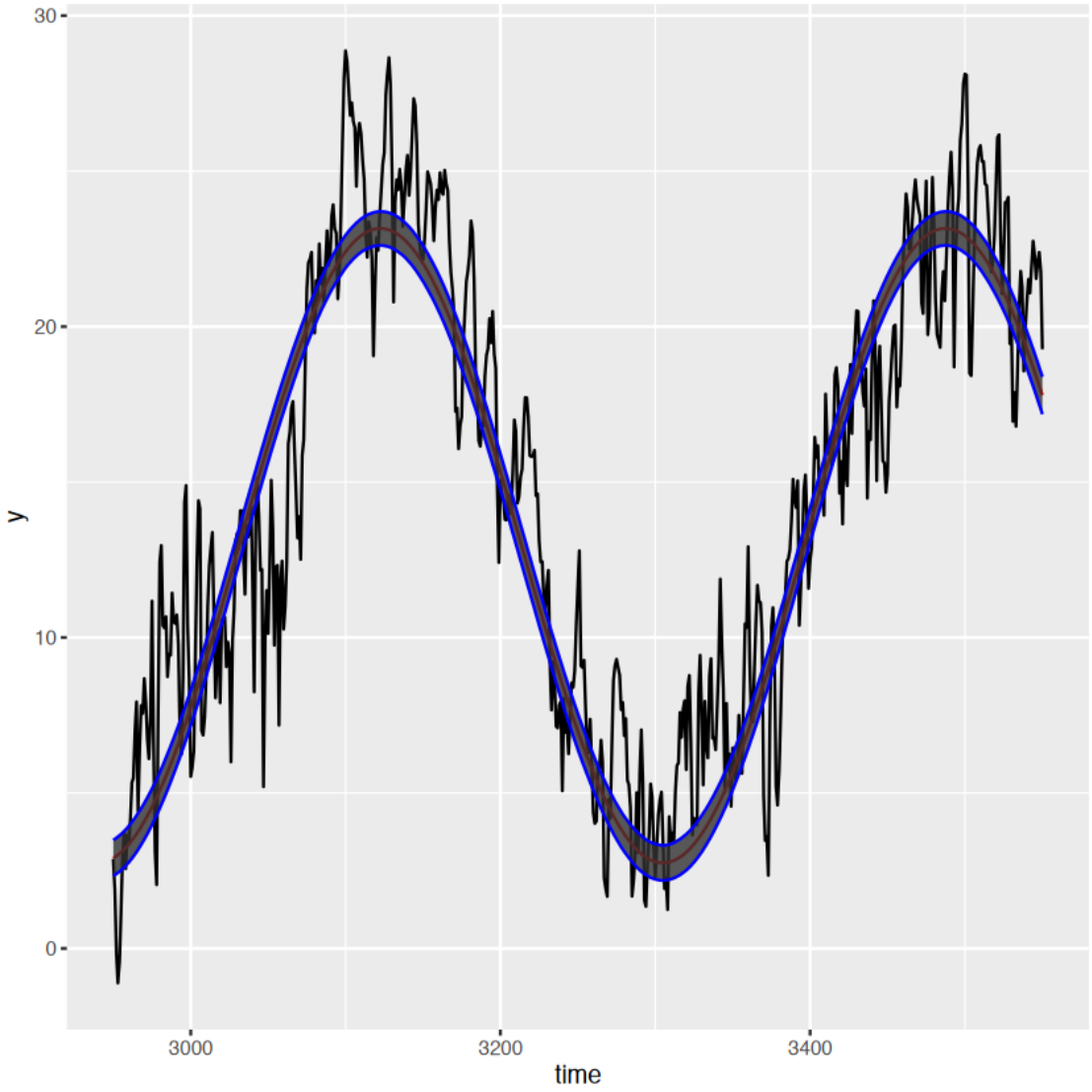
```
Warning message:
"Removed 301 rows containing missing values or values outside the scale
range
(`geom_line()`)."
Warning message in max(ids, na.rm = TRUE):
"nessun argomento non-mancante al massimo; si restituisce -Inf"
```



Vediamo come la funzione seno ha approssimato i dati, tramite la sua media a posteriori e CI

[42]:
```
nsamp = nrow(samp_pred_2)
sine_func = matrix(NA, nrow = length(y), ncol=nsamp)
for(i in 1:nsamp)
{
  sine_func[, i] <- samples_2$mu[i] + samples_2$A[i] * sin(2 * pi * (1:
 ↪length(y)) / 365 + samples_2$c[i])
}

mean_sine <- apply(sine_func, 1, mean)
q1_sine <- apply(sine_func, 1, function(x) quantile(x, probs = 0.025))
q2_sine <- apply(sine_func, 1, function(x) quantile(x, probs = 1-0.025))

data_plot_tot_2 <- data_plot_tot_2 %>%
  mutate(
    mean_sine = mean_sine,
    q1_sine = q1_sine,
    q2_sine = q2_sine
  )
```

[51]:
```
data_plot_tot_2 %>% slice(1:nobs)%>%
  slice((length(y) - 600):length(y)) %>%
  ggplot(aes(x = time, y = y)) +
  geom_line() +
  geom_line(aes(y = mean_sine), col = "red") +
  geom_ribbon(aes(ymin = q1_sine, ymax = q2_sine), alpha = 0.8, col="blue")
```



Anche qui calcoliamo i residui

[44]:
```
res_mat_2 <- matrix(NA, ncol = nobs, nrow = nrow(samples_2$alpha))

for (isim in 1:nrow(samples$alpha))
{
  res_mat_2[isim, 1] <- y[1] - samples$mu[isim] - samples_2$A[isim] * sin(2 *␣
 ↪pi * 1 / 365 + samples_2$c[isim])
  for (i in 2:nobs)
  {
```

```
    res_mat_2[isim, i] <- y[i] - samples$mu[isim] - samples_2$A[isim] * sin(2 *␣
 ↪pi * i / 365 + samples_2$c[isim]) - samples$alpha[isim] * (y[i - 1] -␣
 ↪samples$mu[isim] - samples_2$A[isim] * sin(2 * pi * (i-1) / 365 +␣
 ↪samples_2$c[isim]))
  }
}
```

[45]:
```
par(mfrow = c(1, 2))
acf(colMeans(res_mat_2))
plot(density(colMeans(res_mat_2)))
par(mfrow = c(1, 1))
```