Moving Averages in R

William Chiu

5/7/2022

Data and Noise

Suppose y is a linear function of x_1 and random error:

$$y = \beta_0 + \beta_1 x_1 + \epsilon$$

The linear relationship is simulated below with deterministic values for x_1:

```
set.seed(1)

nobs <- 1000

x1 <- log(1:nobs)
y <- 2 + 3 * x1 + rnorm(nobs)

df <- data.frame(t=1:nobs, y=y, x1=x1)

rm(x1, y)
```

But also suppose that x_1 is not directly observable, instead we observe its noiser cousin that contains random errors. For example, the instrument that measures x_1 suffers from random imprecision.

```
df$x1_noisy <- df$x1 + rnorm(nobs,0)
```

Since x_1 is observed with noisy imprecision, this degrades the OLS fit.

```
true_model <- lm(y ~ x1, data=df)

noisy_model <- lm(y ~ x1_noisy, data=df)
```

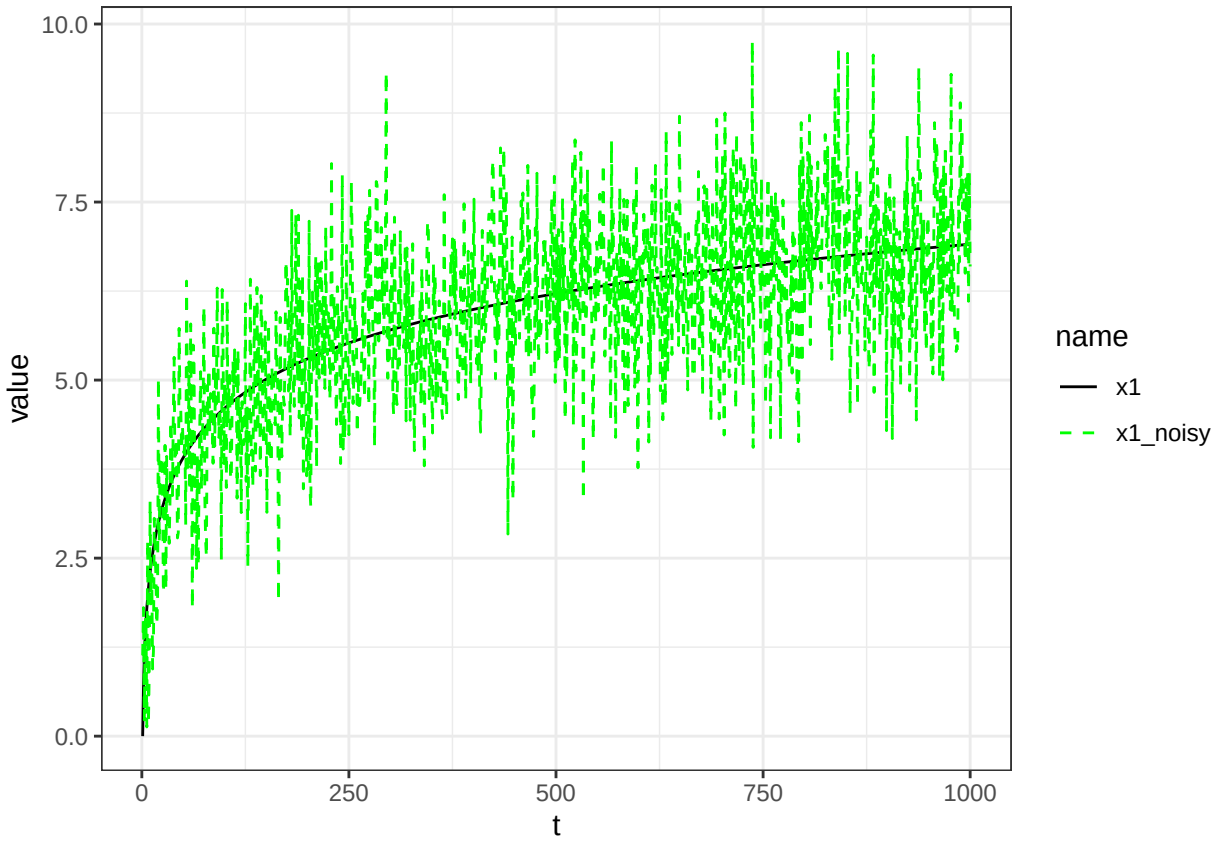
	Dependent variable:	
	y	
	(1)	(2)
Constant	2.203	11.435
	(0.199)***	(0.314)***
x1	2.964	
	(0.033)***	
x1_noisy		1.406
		(0.052)***
Observations	1,000	1,000
R^2	0.889	0.425
Adjusted R^2	0.888	0.425
Residual Std. Error (df = 998)	1.035	2.350
Note:	*p<0.1; **p<0.05; ***p<0.01	

Visualization

Plot x_1 and x_1^{noisy} over time.

```
df_long <- pivot_longer(df, -t) %>% filter(name %in% c('x1','x1_noisy'))

ggplot(df_long, aes(x=t, y=value, group=name, color=name)) +
  geom_line(aes(linetype=name)) +
  scale_linetype_manual(values=c("solid", "dashed"))+
  scale_color_manual(values=c('black', 'green')) +
  theme_bw()
```



Filter and Moving Averages

We could apply a filter on x_1^{noisy} to remove some of the "jumpiness". First, we apply a backward-looking moving average with a k-period window:

```
k_param <- 12

df$x1_noisy_ma <- stats::filter(x=ts(df$x1_noisy),
                                filter=rep(1/k_param, k_param),
                                method="convolution",
                                sides=1)

knitr::kable(head(df, k_param+5))
```

t	y	x1	x1_noisy	x1_noisy_ma
1	1.373546	0.0000000	1.1349651	NA
2	4.263085	0.6931472	1.8050790	NA
3	4.460208	1.0986123	0.2278347	NA
4	7.754164	1.3862944	1.5970259	NA
5	7.157821	1.6094379	1.6788336	NA
6	6.554810	1.7917595	0.1291106	NA
7	8.325159	1.9459101	2.7567501	NA
8	8.976649	2.0794415	0.1670957	NA

t	y	x1	x1_noisy	x1_noisy_ma
9	9.167455	2.1972246	0.9504711	NA
10	8.602367	2.3025851	3.3007395	NA
11	10.705467	2.3978953	1.8570225	NA
12	9.844563	2.4849066	2.2685309	1.489455
13	9.073607	2.5649494	0.9430121	1.473459
14	7.702472	2.6390573	1.1880934	1.422043
15	11.249082	2.7080502	3.0589599	1.657971
16	10.272833	2.7725887	2.5980418	1.741388
17	10.483450	2.8332133	2.2417849	1.788301

Second, we apply a "centered" moving average that includes both past and future periods.

```
df$x1_noisy_ma_ctr <- stats::filter(x=ts(df$x1_noisy),
                                    filter=rep(1/k_param, k_param),
                                    method="convolution",
                                    sides=2)

knitr::kable(head(df, k_param+5))
```

t	y	x1	x1_noisy	x1_noisy_ma	x1_noisy_ma_ctr
1	1.373546	0.0000000	1.1349651	NA	NA
2	4.263085	0.6931472	1.8050790	NA	NA
3	4.460208	1.0986123	0.2278347	NA	NA
4	7.754164	1.3862944	1.5970259	NA	NA
5	7.157821	1.6094379	1.6788336	NA	NA
6	6.554810	1.7917595	0.1291106	NA	1.489455
7	8.325159	1.9459101	2.7567501	NA	1.473459
8	8.976649	2.0794415	0.1670957	NA	1.422043
9	9.167455	2.1972246	0.9504711	NA	1.657971
10	8.602367	2.3025851	3.3007395	NA	1.741388
11	10.705467	2.3978953	1.8570225	NA	1.788301
12	9.844563	2.4849066	2.2685309	1.489455	1.907237
13	9.073607	2.5649494	0.9430121	1.473459	1.831436
14	7.702472	2.6390573	1.1880934	1.422043	2.236831
15	11.249082	2.7080502	3.0589599	1.657971	2.384128
16	10.272833	2.7725887	2.5980418	1.741388	2.431154
17	10.483450	2.8332133	2.2417849	1.788301	2.603110

The filters created `NA` values that should be removed before refitting the models.

```
df_filtered <- df %>% drop_na()

knitr::kable(head(df_filtered))
```

t	y	x1	x1_noisy	x1_noisy_ma	x1_noisy_ma_ctr
12	9.844563	2.484907	2.2685309	1.489455	1.907237
13	9.073607	2.564949	0.9430121	1.473459	1.831436

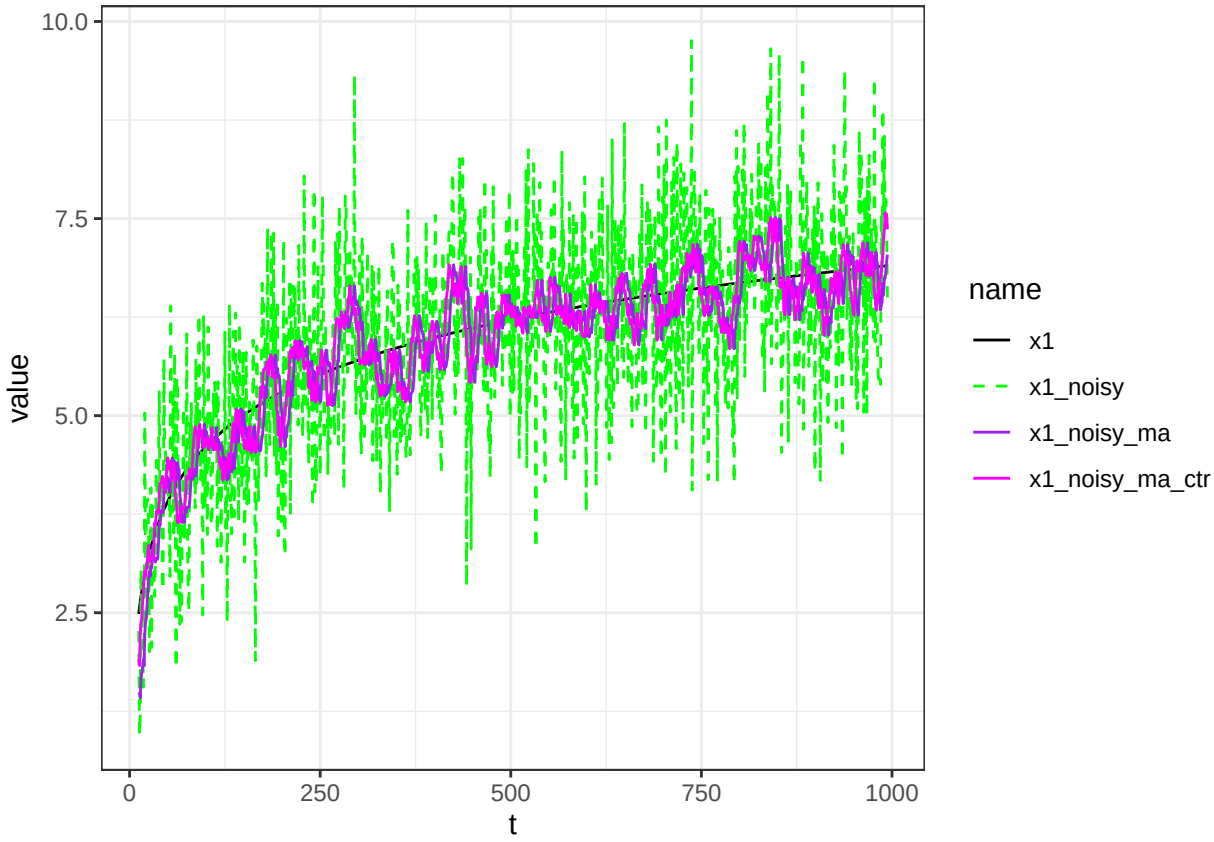
t	y	x1	x1_noisy	x1_noisy_ma	x1_noisy_ma_ctr
14	7.702472	2.639057	1.1880934	1.422043	2.236831
15	11.249082	2.708050	3.0589599	1.657971	2.384128
16	10.272833	2.772589	2.5980418	1.741388	2.431154
17	10.483450	2.833213	2.2417849	1.788301	2.603110

More Visualization

Plot x_1, x_1^{noisy}, and the filtered predictors over time.

```
df_long2 <- pivot_longer(df_filtered, -t) %>%
  filter(name %in%  c('x1','x1_noisy', 'x1_noisy_ma', 'x1_noisy_ma_ctr'))

ggplot(df_long2, aes(x=t, y=value, group=name, color=name)) +
  geom_line(aes(linetype=name)) +
  scale_linetype_manual(values=c("solid", "dashed", "solid", "solid"))+
  scale_color_manual(values=c('black', 'green', 'purple', 'magenta')) +
  theme_bw()
```



Refit models with MA predictors

```r
ma_model <- lm(y ~ x1_noisy_ma, data=df_filtered)

ma_ctr_model <- lm(y ~ x1_noisy_ma_ctr, data=df_filtered)
```

	Dependent variable:			
	y			
	(1)	(2)	(3)	(4)
Constant	2.203	11.435	5.192	4.381
	(0.199)***	(0.314)***	(0.264)***	(0.286)***
x1	2.964			
	(0.033)***			
x1_noisy		1.406		
		(0.052)***		
x1_noisy_ma			2.483	
			(0.044)***	
x1_noisy_ma_ctr				2.604
				(0.048)***
Observations	1,000	1,000	983	983
R^2	0.889	0.425	0.764	0.753
Adjusted R^2	0.888	0.425	0.763	0.753
Residual Std. Error	1.035 (df = 998)	2.350 (df = 998)	1.362 (df = 981)	1.392 (df = 981)
Note:				*p<0.1; **p<0.05; ***p<0.01