DSC 152:
Applied Statistical Data Analysis and Inference

Lecture #18
Time Series Regression

Thursday, May 28
Spring Quarter 2026
Peter Chi

Recall. . .

Conditions for validity of inference in linear regression

- The relationship between X and Y, if there is one, is actually **L**inear
  - e.g. not quadratic, exponential, etc.
- **I**ndependence of observations
- **N**ormality of ϵ_i
  - Note that this can also be achieved, due to the central limit theorem, with a large sample size even if ϵ_i does not follow a normal distribution
- **E**qual variance across all values of X
  - Also known as homoskedasticity

Time Series Data

Last time:

Auto-regressive Models: AR(p)

Current observations are a function of p previous steps. Result: shocks to the system linger for many steps.

- ex: stock prices

Moving Average Models: MA(q)

Current observations are a function of the *deviation* from the average of q previous steps. Result: shocks to the system are completely gone after q steps.

- ex: equipment malfunction in a shop that can be quickly fixed

Time Series Data

Mathematically:

An AR(p) model:

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + \epsilon_t$$

(see Lab 9 for more details)

An MA(q) model:

$$y_t = c + \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \cdots + \theta_q \epsilon_{t-q}$$

where each ϵ_i is independently $N(0, \sigma^2)$.

Let's break this one down a little here...

AR(p) vs. MA(q) models

But first: what is the difference between these two?

For simplicity, let's consider AR(1) vs. MA(1) with $c = 0$ (no drift).

AR(1)

$$y_t = \phi_1 y_{t-1} + \epsilon_t$$

MA(1)

$$y_t = \epsilon_t + \theta_1 \epsilon_{t-1}$$

Key difference:

- In the AR(1) model, even though y_t only explicitly depends on y_{t-1}, it is also true that y_{t-1} depended on y_{t-2}, and so on. So, impacts last longer than just the value of p.
- Conversely, in the MA(1) model, y_t depends on y_{t-1} only through its error term. So, impacts only last the length of q.

MA(q) models

Again, suppose $q = 1$. Then, an MA(1) model would be:

$$y_t = c + \epsilon_t + \theta_1 \epsilon_{t-1}$$

So as a simple example, suppose $c = 0, \theta_1 = 1, \sigma = 1$. Then ϵ_1 and ϵ_0 are both independently $N(0, 1)$ so,

```
set.seed(1)
eps0 <- rnorm(n=1, mean=0, sd=1)
eps1 <- rnorm(n=1, mean=0, sd=1)
eps0
```

```
## [1] -0.6264538
```

```
eps1
```

```
## [1] 0.1836433
```

MA(q) models

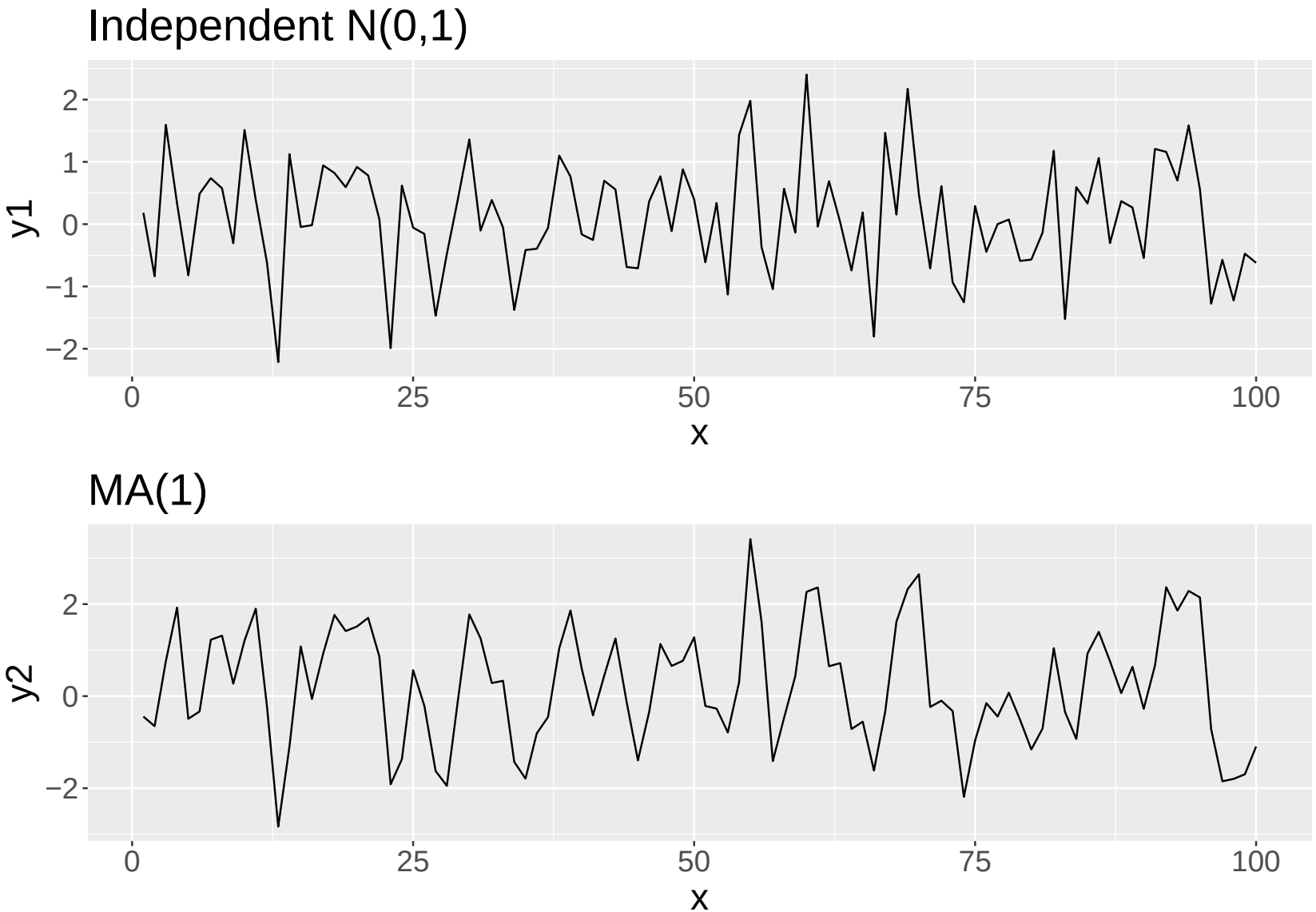
Now,

$$
\begin{aligned}
y_1 &= \epsilon_1 + \epsilon_0 = 0.1836433 + -0.6264538 = -0.4428105 \\
y_2 &= \epsilon_2 + \epsilon_1 = -0.8356286 + 0.1836433 = -0.6519853 \\
y_3 &= \epsilon_3 + \epsilon_2 = 1.5952808 + -0.8356286 = 0.7596522 \\
y_4 &= \epsilon_4 + \epsilon_3 = 0.3295078 + 1.5952808 = 1.9247886 \\
y_5 &= \epsilon_5 + \epsilon_4 = -0.8204684 + 0.3295078 = -0.4909606 \\
y_6 &= \epsilon_6 + \epsilon_5 = 0.4874291 + -0.8204684 = -0.3330393
\end{aligned}
$$

ok but what is actually happening here?

MA(q) models

Comparing the MA(1) model to a simple independent N(0, 1) model:



MA(q) models

Here's the code that made the plot on the previous slide:

```
library(ggplot2)
library(gridExtra)
set.seed(1)
theme_update(text=element_text(size=20))
x <- 1:100
y1 <- rnorm(101) # 1 extra for "burn-in"
y2 <- NULL
for(i in 1:100){
  y2[i] <- y1[i+1] + y1[i]
}

dat1 <- data.frame(x=x, y1=y1[2:101])
dat2 <- data.frame(x=x, y2=y2)

p1 <- ggplot(data=dat1, aes(x=x, y=y1)) + geom_line() +
  labs(title="Independent N(0,1)")
p2 <- ggplot(data=dat2, aes(x=x, y=y2)) + geom_line() +
  labs(title="MA(1)")

grid.arrange(p1, p2, ncol=1)
```

MA(1) model vs. AR(1) model

Let's quickly go back and compare...

In the MA(1) model we had:

$$y_1 = \epsilon_1 + \epsilon_0$$
$$y_2 = \epsilon_2 + \epsilon_1$$
$$y_3 = \epsilon_3 + \epsilon_2$$

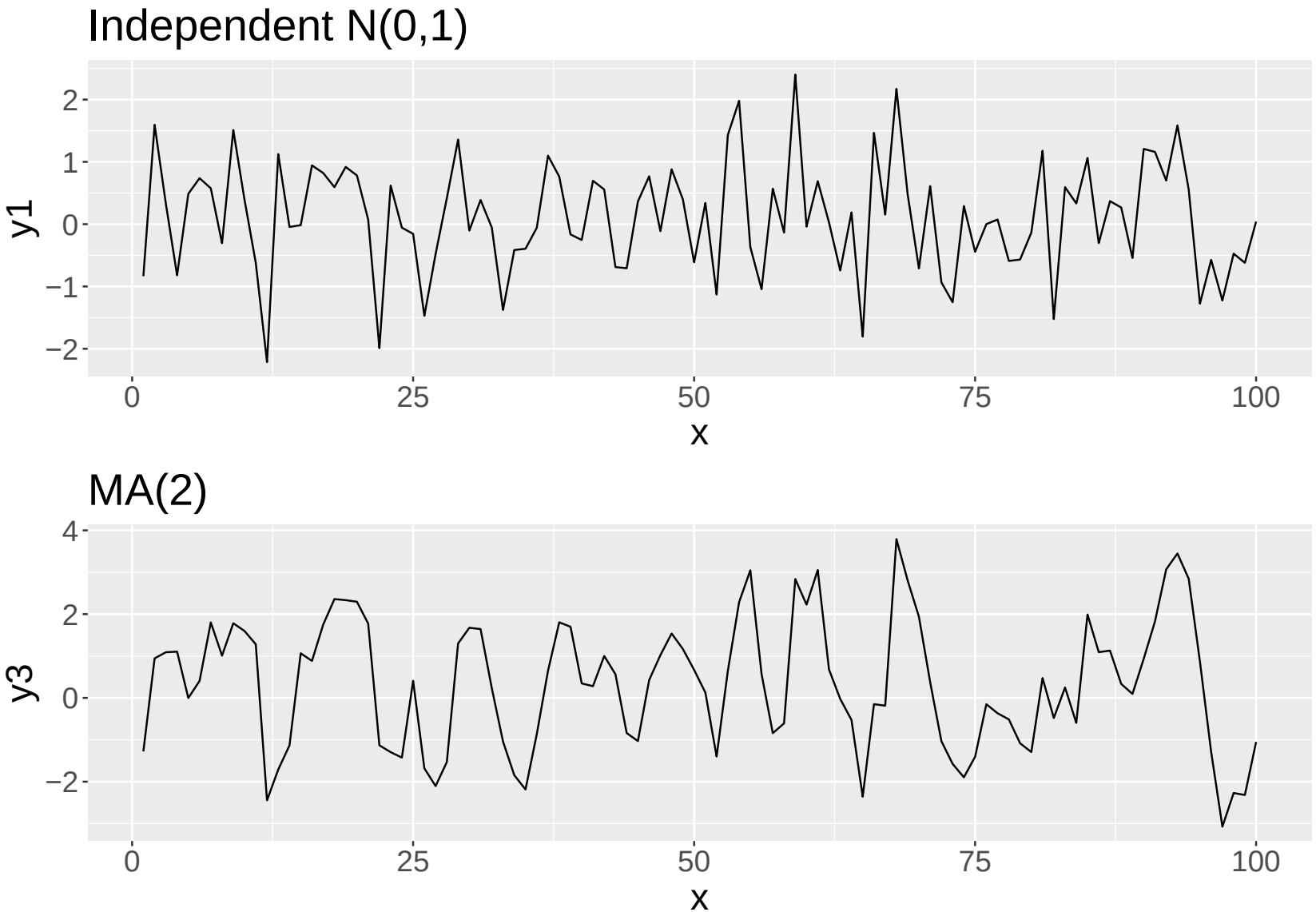
Conversely, in the AR(1) model,

with $\phi_1 = 1$ and suppose $y_0 = 0$, it would be:

$$y_1 = y_0 + \epsilon_1 = \epsilon_1$$
$$y_2 = y_1 + \epsilon_2 = y_0 + \epsilon_1 + \epsilon_2$$
$$y_3 = y_2 + \epsilon_3 = y_0 + \epsilon_1 + \epsilon_2 + \epsilon_3$$

MA(q) models

Now comparing the MA(2) model to a independent N(0, 1) model:



MA(q) models

Your Turn #1

- Follow the code on Slide 9 to plot data from an MA(10) model below data from an independent N(0,1) model. Use the same specifications as in the code on Slide 9.

- Recall the behavior of vectors and their indices, to make at least one part of this slightly less cumbersome to code.

- Comment briefly on what you observe.

Lucas 2013 paper and Lab 9

Time Series Regression

Now, how does a time series behavior of data fit into the regression setting?

Recall: Simple Linear Regression

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i$$

where each $\epsilon \sim N(0, \sigma^2)$.

In general, Time Series regression fixes the situation where each of the y_is are not independent! It does so through the ϵ_i values.

Regression with a single X covariate and an MA(1) term

$$y_t = \beta_0 + \beta_1 x_t + \eta_t$$

where $\eta_t = \epsilon_t + \theta_1 \epsilon_{t-1}$.

To keep things simple(r), first we will simulate data according to the following situation:

- We are interested in modeling a particular NBA basketball player's points per game.
  - We suspect that the opponent's defensive rating influences the player's points per game. A team's defensive rating is basically the average number of points the team allows per 100 possessions; a lower defensive rating indicates that their defense is better!

- Let's assume that the subsequent opponent will adjust to the player's success in any given game (i.e. fit an MA(1) term)
  - If the player has a high scoring game, the next opponent will tend to double-team him

Simulation setup

- X is the current opponent's end of season defensive rating (ranging from approximately 106 to 121)
- $\beta_0 = -38, \beta_1 = 0.55$ – baseline of -38 points, with 0.55 more points per every 1-unit increase in opponent's defensive rating
- $\theta_1 = -0.55$ (55% of the previous game's deviation from the average carries inversely into the current game)
- $\sigma = 8$ (typical random daily variation in points scored by this player)

Time Series Regression

Simulated Example: NBA player scoring vs. Opponent's Defensive Rating

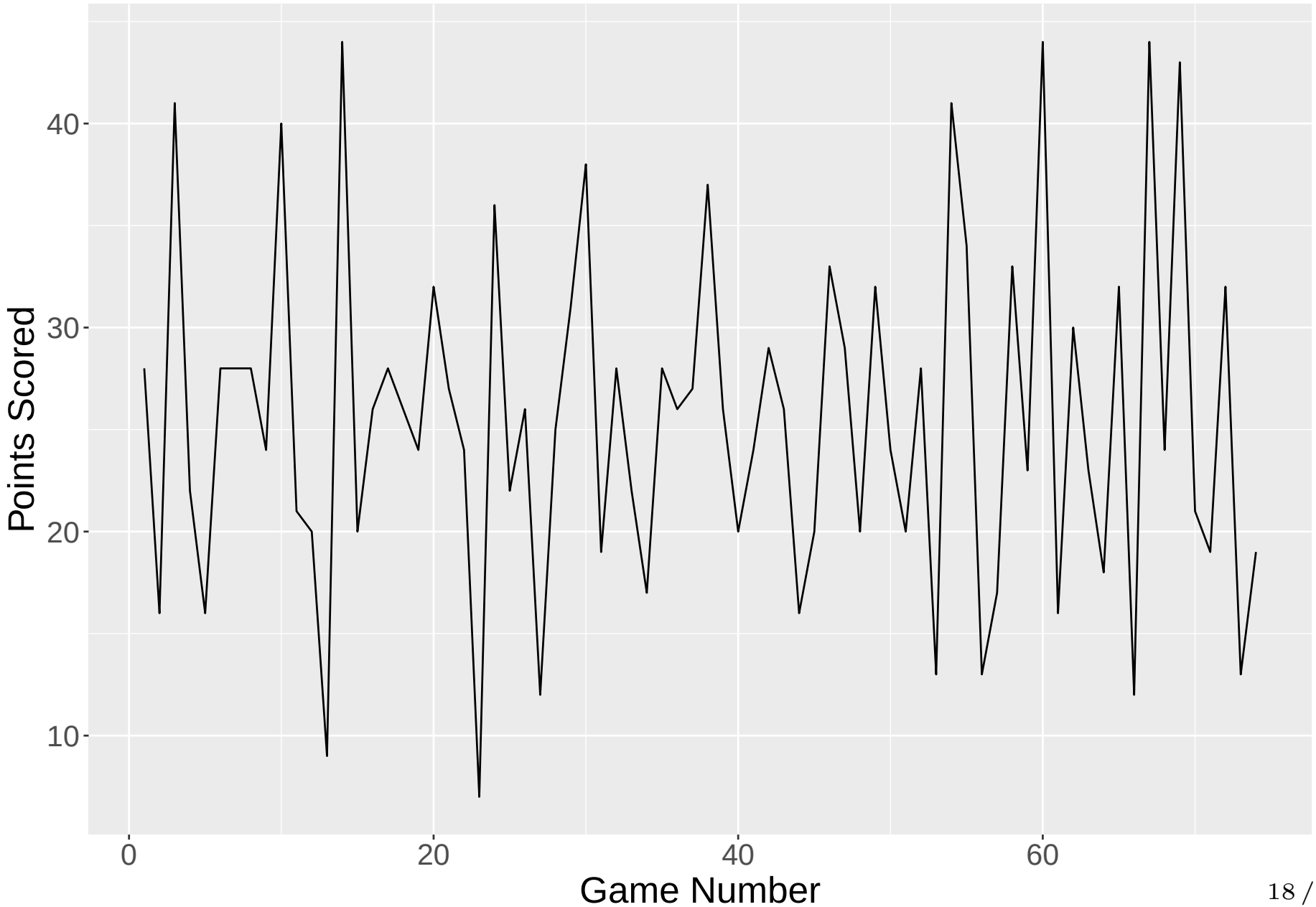
```
library(readr)
x <- read_csv("JBrunson25-26.csv")$DefRtg
set.seed(1)
eps <- rnorm(length(x)+1, mean=0, sd=8)
eta <- NULL
for(i in 1:length(x)){
  eta[i] <- eps[i+1] - 0.55*eps[i]
}

pts <- round(-38 + 0.55*x + eta, 0)

nba_df <- data.frame(pts=pts,
                     opp_def=x,
                     game_num = 1:length(x))
```

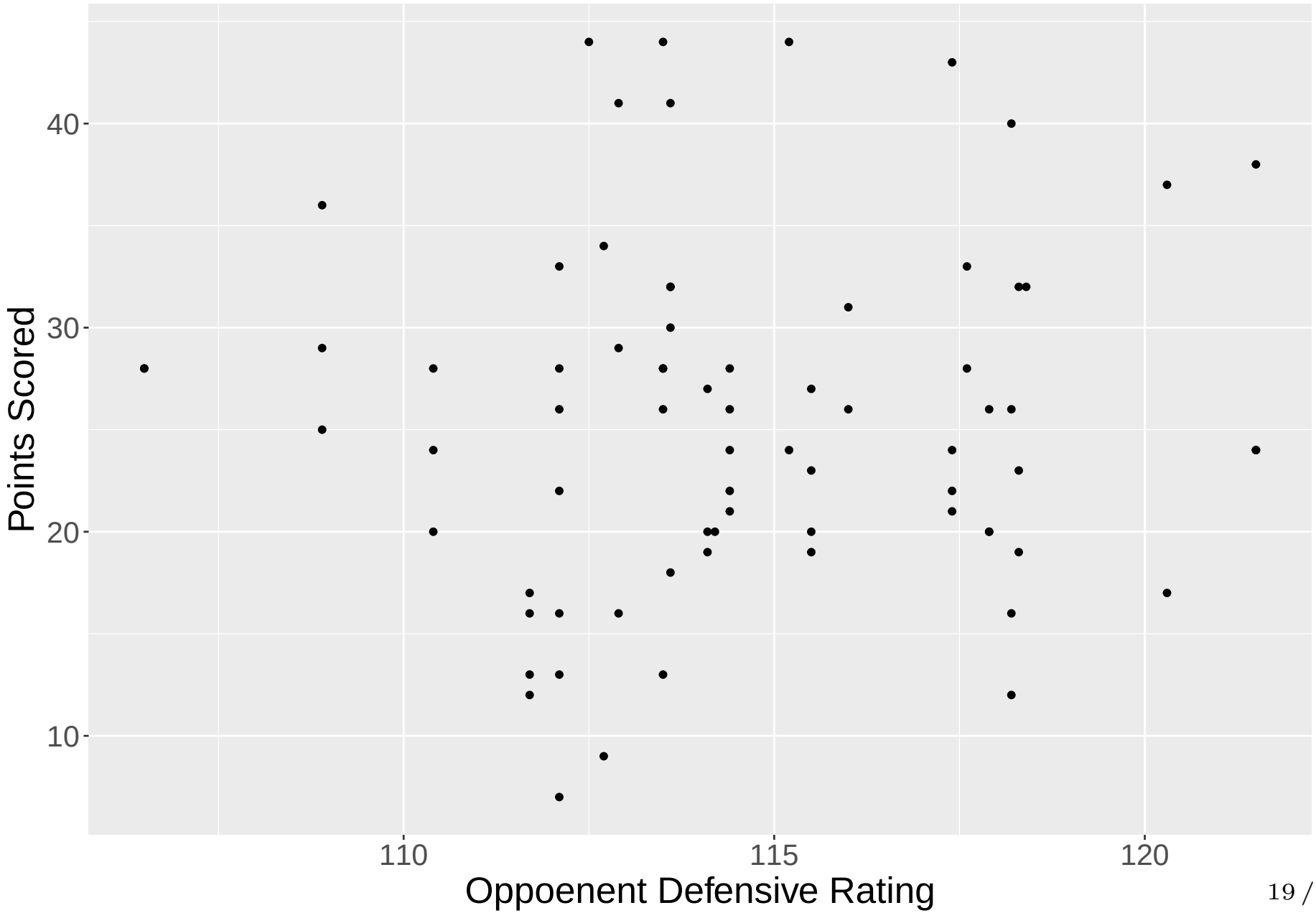
Time Series Regression

Simulated Example: NBA player scoring vs. Opponent's Defensive Rating



Time Series Regression

Simulated Example: NBA player scoring vs. Opponent's Defensive Rating



Time Series Regression

Simulated Example: NBA player scoring vs. Opponent's Defensive Rating

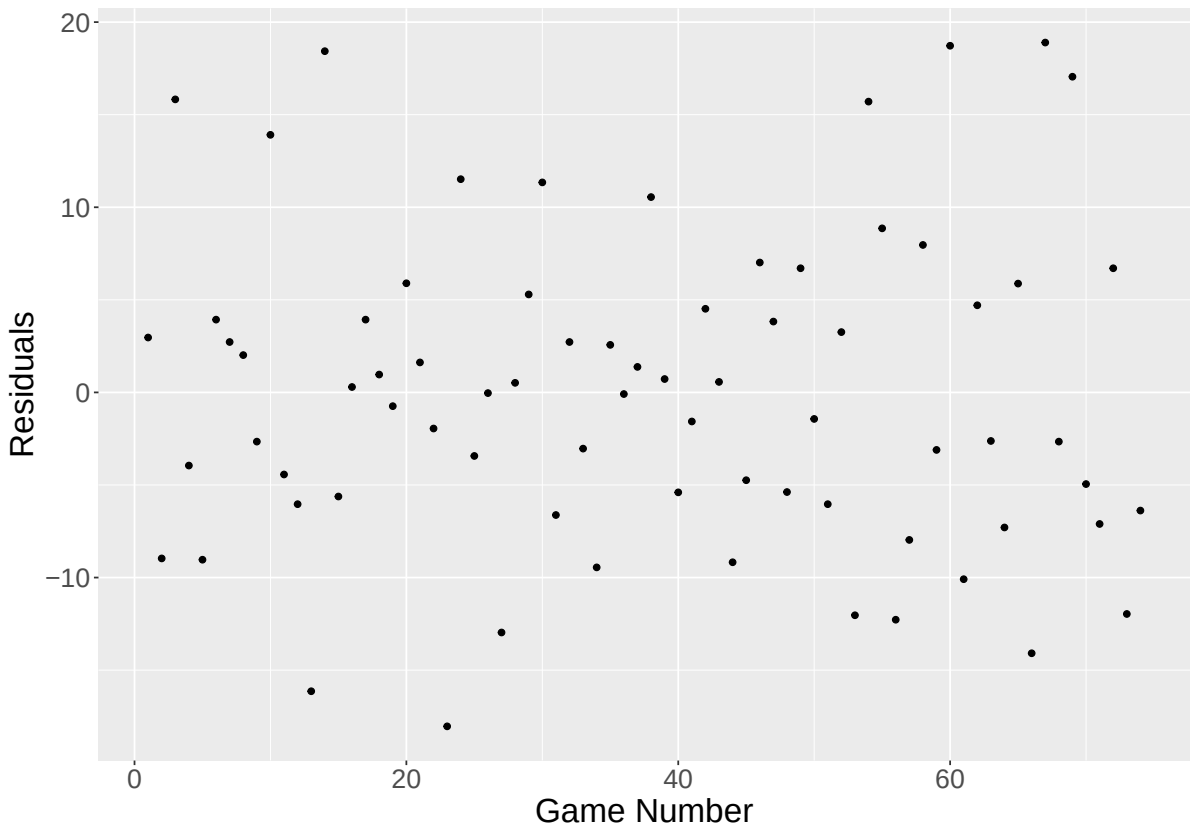
```r
model1 <- lm(pts ~ opp_def, data=nba_df)
summary(model1)
```

```
##
## Call:
## lm(formula = pts ~ opp_def, data = nba_df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -18.0358  -5.9330  -0.0625   4.6581  18.8952
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   5.6929    34.8574   0.163    0.871
## opp_def       0.1726     0.3042   0.567    0.572
##
## Residual standard error: 8.583 on 72 degrees of freedom
## Multiple R-squared:  0.00445,    Adjusted R-squared:  -0.009377
## F-statistic: 0.3218 on 1 and 72 DF,  p-value: 0.5723
```

Time Series Regression

Simulated Example: NBA player scoring vs. Opponent's Defensive Rating

```
nba_df$residuals <- model1$residuals
ggplot(nba_df, aes(x=game_num, y=residuals)) +
  geom_point() +
  labs(y="Residuals", x="Game Number")
```



Now, if we want to properly account for the MA(1) structure:

This is the model we need to fit:

$$y_t = \beta_0 + \beta_1 x_t + \eta_t$$

where $\eta_t = \epsilon_t + \theta_1 \epsilon_{t-1}$.

The `Arima` function in the `forecast` package is the way to do it!

The `Arima` function:

```
library(forecast)
model2 <- Arima(
  y = nba_df$pts,
  order = c(0, 0, 1),
  xreg = nba_df$opp_def)
```

where

- `y` is the outcome variable.
- `order` is to specify that this is an AR(p) or MA(q) model:
  - The first value is the p for AR(p).
  - The second value is a "differencing" term – we will not use that.
  - The third value is the q for MA(q).
- `xreg` is the "design matrix," or basically a matrix whose columns are each one of the predictor variables of interest.

along with `coeftest` from the `lmtest` package:

```
library(lmtest)
coeftest(model2)
```

```
##
## z test of coefficients:
##
##              Estimate Std. Error z value  Pr(>|z|)
## ma1          -0.808975   0.081002 -9.9871 < 2.2e-16 ***
## intercept -28.000936  17.254307 -1.6228  0.104624
## xreg          0.467400   0.150572  3.1042  0.001908 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 '
```

A few key points

- We observe that the model that properly accounts for the MA(1) structure results in us correctly rejecting H_0, whereas the simple linear model misses it.
- The time series trend is perhaps somewhat observable in the line plot over time, but does not show up at all in the plot of residuals over time
- The structure of inputs to the `Arima` function is unfortunately a bit awkward...

For the last point, we'll see how we have to handle that in the next exercise.

Now for real data

- The opponents' defensive ratings came from a dataframe of NBA player Jalen Brunson's game statistics from the regular season of this past year (2025-2026).

- But in the previous example, we simulated the points scored in each game so that we could look at what happens when we definitively know that the data followed an MA(1) structure. Let's now examine the real data.

The full dataset is in the `JBrunson25-26.csv` file on the course website. Each row of the dataframe represents a single game.

Time Series Regression

Real Data: Jalen Brunson scoring vs. Opponent's Defensive Rating

Our primary question of interest is still whether the opponent's defensive rating is associated with the number of points that Brunson scores in each game, and we still would like to model it with an MA(1) term. Suppose we would also like to adjust for the following covariates:

- `MIN`: the number of minutes he played in that game
- `TO`: the number of turnovers that he had in that game
- `Location`: whether the game was `Home` or `Away`.

Your Turn #2

Real Data: Jalen Brunson scoring vs. Opponent's Defensive Rating

- Create a matrix `X` whose columns consist of the values of each covariate that you want to put in the model (including the primary covariate!)

- Run the `Arima` function with the appropriate inputs to perform a time series regression on points scored, with your `X` matrix created above containing your covariates for the model, and with an MA(1) term.
  - Note that, at first, it will be likely unhappy with your `X` matrix. Try to fix that based on the error message that it gives.

- As sensitivity analyses,
  - Plot `PTS` over each game in order
  - Run a multiple linear model with the same covariate as above, but without the MA(1) term
  - Plot the residuals from this model over time

- Briefly comment on what you observe across all parts.

Summary and Recap

Summary

- AR(p) and MA(q) are two different models to account for time series data. They have similarities and differences.
- To fit a time series regression with AR(p) and MA(q) terms, we use the `Arima` function. Correctly providing the input variables to the `Arima` function can be a bit of a hassle.

Looking Ahead

Consequences of model misspecification in time series data

Today's Daily Check

The two Your Turns