Daniel Losada and Sergio Quintanilla

```r
library(carData) #to load dataset
library(dplyr)
library(MASS)
```

Exercise 1

a) LM Standard

```r
model = lm(prestige~ income+education, Duncan)
n = nrow(Duncan)
k = 2 #2 or 3? not 100% sure
y_hat = model$fitted.values
y_bar = mean(Duncan$prestige)
E_hat = model$residuals

F_stat_func = function(y_hat,y_bar,E_hat,n,k){
  numerator = sum(((y_hat - y_bar)^2)/k)
  denominator = sum((E_hat^2)/(n-k-1))
  F_statistic = numerator/denominator
  F_statistic
}
F_stat_func(y_hat,y_bar,E_hat,n,k)
```

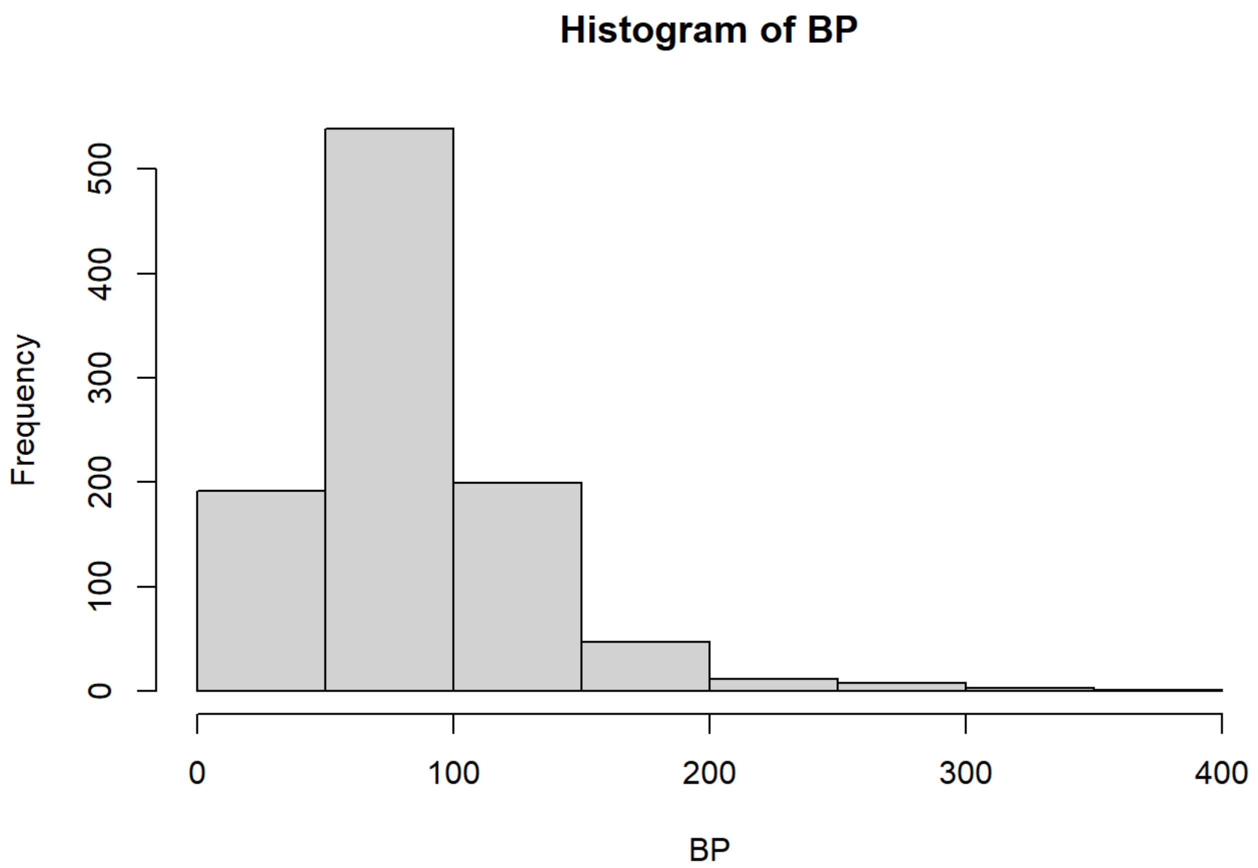
```
## [1] 101.2162
```

b) Bootstrap Pairs

```r
N = 1000 #number of times to repeat the resampling and recalc shit

B_pairs = function(N){
  out = vector(length=N)
  for (i in 1:N){
      samp = sample_n(Duncan,30,replace=T)
      model = lm(prestige~ income+education, samp)
      n = nrow(samp)
      k = 2
      y_hat = model$fitted.values
      y_bar = mean(samp$prestige)
      E_hat = model$residuals
      out[i] = F_stat_func(y_hat,y_bar,E_hat,n,k)
  }
  out
}

BP = B_pairs(N)
hist(BP)
```




```
sd(BP) / sqrt(length(BP)) #standard error
```

```
## [1] 1.413319
```

c) Bootsrap Residuals

```
N = 1000 #number of times to repeat the resampling and recalc

B_residuals = function(N){
  out = vector(length=N)
  for (i in 1:N){
      E_samp = sample(E_hat,30,replace=T)
      Y_hat_star = y_hat + E_samp
      out[i] = F_stat_func(Y_hat_star,y_bar,E_samp,n,k)
  }
  out
}

BR = B_residuals(N)
hist(BR)
```


```
sd(BR) / sqrt(length(BR)) #standard error
```

```
## [1] 1.851912
```

d) RLM Standard

```
model = rlm(prestige~ income+education, Duncan)
summary(model)
```

```
##
## Call: rlm(formula = prestige ~ income + education, data = Duncan)
## Residuals:
##     Min      1Q  Median      3Q     Max
## -30.120  -6.889   1.291   4.592  38.603
##
## Coefficients:
##             Value   Std. Error t value
## (Intercept) -7.1107  3.8813    -1.8320
## income       0.7014  0.1087     6.4516
## education    0.4854  0.0893     5.4380
##
## Residual standard error: 9.892 on 42 degrees of freedom
```

```
n = nrow(Duncan)
k = 2 #2 or 3? not 100% sure
y_hat = model$fitted.values
y_bar = mean(Duncan$prestige)
E_hat = model$residuals
F_stat_func(y_hat,y_bar,E_hat,n,k)
```

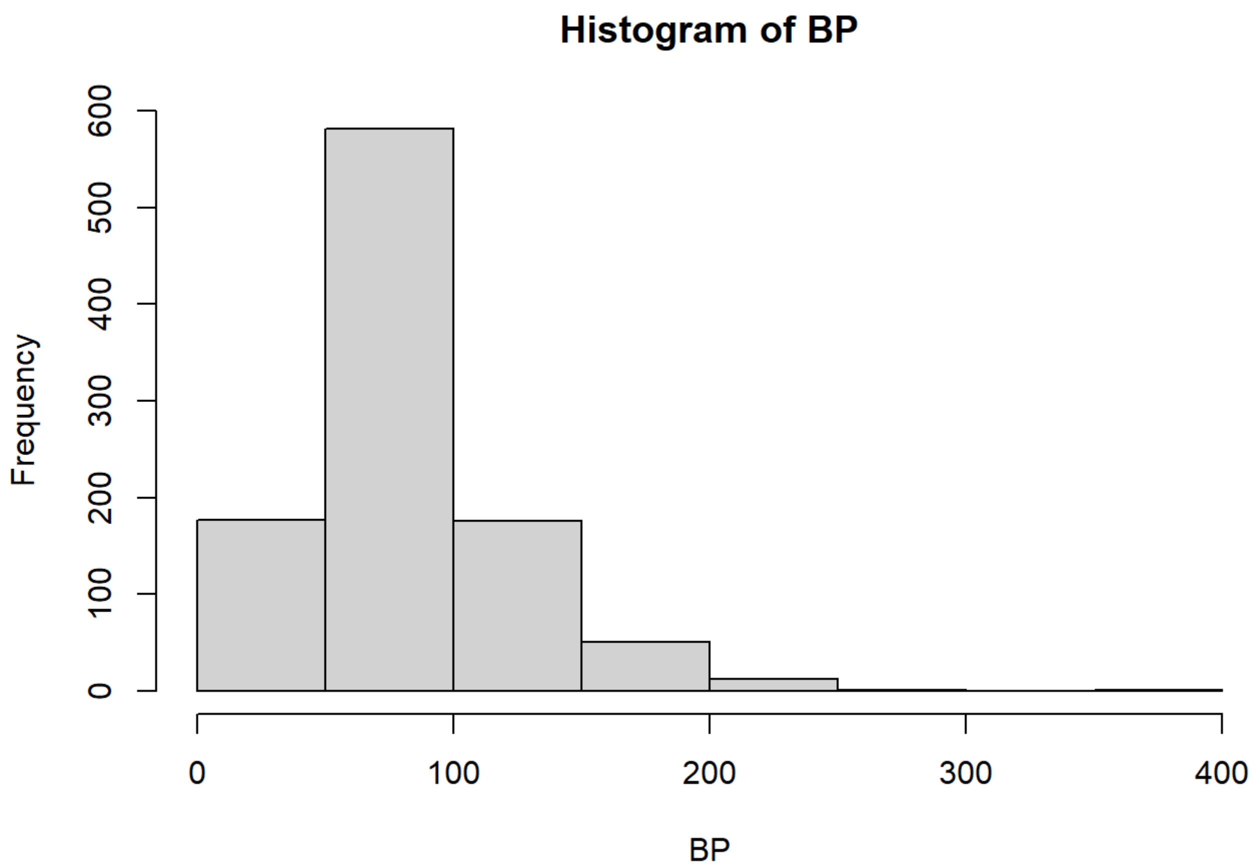
```
## [1] 104.1872
```

RLM with Bootstrap Pairs

```
N = 1000 #number of times to repeat the resampling and recalc shit

B_pairs_rlm = function(N){
  out = vector(length=N)
  for (i in 1:N){
      samp = sample_n(Duncan,30,replace=T)
      model = rlm(prestige~ income+education, samp)
      n = nrow(samp)
      k = 2
      y_hat = model$fitted.values
      y_bar = mean(samp$prestige)
      E_hat = model$residuals
      out[i] = F_stat_func(y_hat,y_bar,E_hat,n,k)
  }
  out
}

BP = B_pairs_rlm(N)
hist(BP)
```




```
sd(BP) / sqrt(length(BP)) #standard error
```

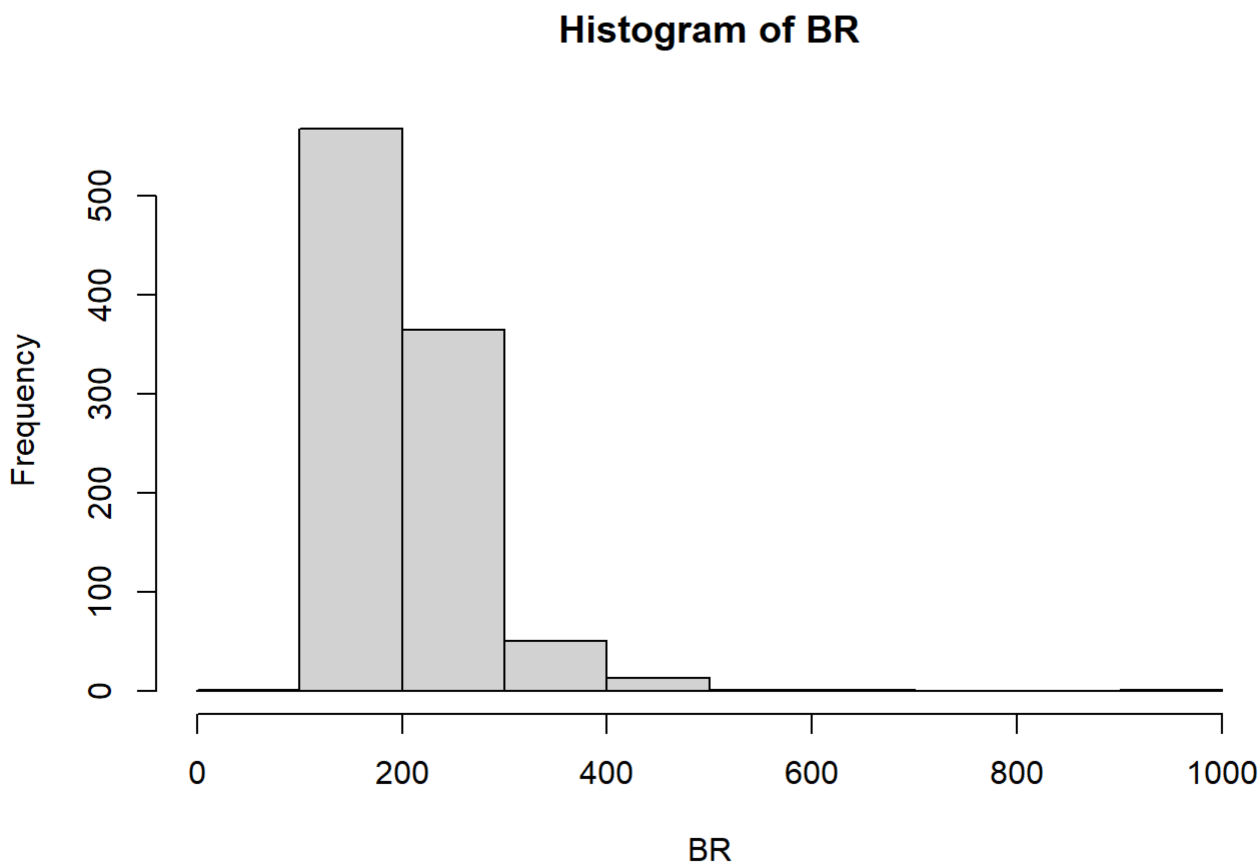
```
## [1] 1.22623
```

RLM with Bootstrap Residuals

```
N = 1000 #number of times to repeat the resampling and recalc

B_residuals = function(N){
  out = vector(length=N)
  for (i in 1:N){
      E_samp = sample(E_hat,30,replace=T)
      Y_hat_star = y_hat + E_samp
      out[i] = F_stat_func(Y_hat_star,y_bar,E_samp,n,k)
  }
  out
}

BR = B_residuals(N)
hist(BR)
```




```
sd(BR) / sqrt(length(BR)) #standard error
```

```
## [1] 2.129495
```

The best performance we saw was in both bootstrap pairs approaches, most notably the rlm models, although both were quite good. This makes sense considering the operation time to accuracy tradeoff we see in most resampling and simulation situations. BP samples generally showed smaller F statistic standard errors and we can see in their histograms they had less spread than the BR samples. However, BR executed far faster than BP samples, for not a huge loss in accuracy. If we needed to work with vastly larger resampling sizes more frequently, BR sampling might be more effective.

Exercise 2

2.1. Boot library

```
# Load necessary libraries
library(stats)
library(boot)
library(bootstrap)

# Set seed for reproducibility
set.seed(123)

# Simulate ARMA(1,1) process
n = 1200
ar.params = 0.6  # AR(1) coefficient
ma.params = -0.5  # MA(1) coefficient

sim_data = arima.sim(n=n, list(ar=ar.params, ma=ma.params))

# Fit ARMA(1,1) model using Maximum Likelihood Estimation (MLE)
model = arima(sim_data, order=c(1,0,1))

# Extract estimated parameters
param_estimates = coef(model)

arima_fit_resid = function(data, indices) {
  boot_sample = data[indices]
  fit = arima(boot_sample, order=c(1,0,1), optim.control = list(maxit=500))
  return(coef(fit))
}

# Residual bootstrap
set.seed(1)
boot_res = boot(sim_data, statistic=arima_fit_resid, R=1000)
valid_results = boot_res$t[complete.cases(boot_res$t), ]
resid_std_errors = apply(valid_results, 2, sd)

arima_fit_resid_block = function(ts_data) {
  fit = tryCatch(
    arima(ts_data, order=c(1,0,1), optim.control = list(maxit=500)),
    error = function(e) return(rep(NA, 3))  # Return NA if estimation fails
  )
  return(coef(fit))
}

# Block bootstrap with different block sizes
block_sizes = c(5, 10, 20)
block_bootstrap_results = list()

for (l in block_sizes) {
  block_fit = tsboot(sim_data, statistic=arima_fit_resid_block, R=1000, l=l, sim="fixed")
  #block_bootstrap_results[[as.character(l)]] = apply(block_fit$t, 2, sd)
  block_bootstrap_results[[as.character(l)]] = apply(block_fit$t, 2, sd)
}

# Print results
print("MLE Estimates:")
```

```
## [1] "MLE Estimates:"
```

```
print(param_estimates)
```

```
##         ar1         ma1   intercept
##  0.47258688 -0.41755068  0.02508379
```

```
print("Residual Bootstrap Standard Errors:")
```

```
## [1] "Residual Bootstrap Standard Errors:"
```

```
print(resid_std_errors)
```

```
## [1] 0.60936560 0.61190056 0.02866819
```

```
for (l in block_sizes) {
  print(paste("Block Bootstrap Standard Errors (Block size =", l, "):"))
  print(block_bootstrap_results[[as.character(l)]])
}
```

```
## [1] "Block Bootstrap Standard Errors (Block size = 5 ):"
## [1] 0.51748842 0.51812073 0.03038005
## [1] "Block Bootstrap Standard Errors (Block size = 10 ):"
## [1] 0.48756691 0.48543202 0.03009119
## [1] "Block Bootstrap Standard Errors (Block size = 20 ):"
## [1] 0.48199535 0.48092410 0.03125521
```

Analysis of Bootstrap Results for ARMA(1,1) Model (R=1000)

MLE Estimates (Baseline Estimates)

The Maximum Likelihood Estimates (MLE) for the ARMA(1,1) parameters are: | Parameter | Estimate | | ————|———–| | **AR(1)** | 0.4726 | | **MA(1)** | -0.4176 | | **Intercept** | 0.0251 |

These serve as the **baseline estimates** for comparison with the bootstrap standard errors.

Residual Bootstrap Standard Errors

Parameter	Residual Bootstrap SE
AR(1)	0.6094
MA(1)	0.6119
Intercept	0.0287

Key Observations:

- **Highest standard errors** among all methods.
- This method **assumes i.i.d. residuals**, which is **not ideal for time series** since it ignores dependence.
- Likely **overestimates uncertainty**, as it does not preserve time structure.

Block Bootstrap Standard Errors (Capturing Dependence)

Parameter	SE (l=5)	SE (l=10)	SE (l=20)
AR(1)	0.5175	0.4876	0.4820
MA(1)	0.5181	0.4854	0.4809
Intercept	0.0304	0.0301	0.0313

Key Observations:

- **Block bootstrap SEs are consistently lower than residual bootstrap SEs.**
- **Block size `l=5` may underestimate variance**, as its SEs are slightly larger than `l=10` and `l=20`.
- **SEs stabilize at `l=10` and `l=20`, suggesting they capture dependence better.**

Final Recommendation

- **Use Block Bootstrap with `l=10`** for reporting standard errors, as it provides stable estimates without excessive smoothing.
- **Avoid residual bootstrap**, as it assumes independence and likely overestimates uncertainty.
- **Block bootstrap SEs provide a more accurate representation of parameter variability in time series models.**

2.2. By hand implementation

```
# -------------------------------
# Residual Bootstrap "By Hand"
# -------------------------------

orig_estimates <- coef(model)

# Extract residuals (et) and construct fitted values (xt_hat = xt - et)
residuals_orig <- residuals(model)
fitted_vals <- sim_data - residuals_orig

# Loop over R bootstrap replications
R <- 1000
param_matrix <- matrix(NA, nrow=R, ncol=length(orig_estimates))
colnames(param_matrix) <- names(orig_estimates)

set.seed(1)
for(b in 1:R) {
  # (a) Sample residuals with replacement
  res_star <- sample(residuals_orig, size=n, replace=TRUE)

  # (b) Create the bootstrap series
  #     x*_t = x-hat_t + e*_t
  #     i.e. “fitted value” + “sampled residual”
  sim_star <- fitted_vals + res_star

  # (c) Re-fit ARMA(1,1) to the bootstrap series
  fit_star <- arima(sim_star, order=c(1,0,1), optim.control=list(maxit=500))

  # (d) Store the parameter estimates
  param_matrix[b, ] <- coef(fit_star)
}

# Compute standard errors as SD across all replications
resid_boot_std_err <- apply(param_matrix, 2, sd)

# -------------------------------
# Results
# -------------------------------
cat("Original MLE Estimates:\n")
```

```
## Original MLE Estimates:
```

```
print(orig_estimates)
```

```
##         ar1         ma1   intercept
##  0.47258688 -0.41755068  0.02508379
```

```
cat("\nResidual-Bootstrap Std. Errors:\n")
```

```
##
## Residual-Bootstrap Std. Errors:
```

```
print(resid_boot_std_err)
```

```
##        ar1        ma1  intercept
## 0.61793416 0.61930167 0.02806959
```

```
# Manual Block Bootstrap
manual_block_boot = function(ts_data, block_size, R) {
  n = length(ts_data)
  num_blocks = ceiling(n / block_size)
  boot_results = matrix(NA, nrow=R, ncol=3)  # Store bootstrap estimates

  for (i in 1:R) {
    start_points = sample(1:(n-block_size+1), num_blocks, replace=TRUE)  # Sample block start
points
    resampled_series = unlist(lapply(start_points, function(start) ts_data[start:(start+block
_size-1)]))
    resampled_series = resampled_series[1:n]  # Truncate to original length
    new_model = arima(resampled_series, order=c(1,0,1), optim.control=list(maxit=500))  # Ref
it model
    boot_results[i, ] = coef(new_model)
  }

  return(apply(boot_results, 2, sd))  # Return standard errors
}

# Compute standard errors for different block sizes
set.seed(123)
manual_block_SE_5 = manual_block_boot(sim_data, block_size=5, R=1000)
manual_block_SE_10 = manual_block_boot(sim_data, block_size=10, R=1000)
manual_block_SE_20 = manual_block_boot(sim_data, block_size=20, R=1000)

# Print results
print("Manual Block Bootstrap Standard Errors (l=5):")
```

```
## [1] "Manual Block Bootstrap Standard Errors (l=5):"
```

```
print(manual_block_SE_5)
```

```
## [1] 0.50340582 0.50196027 0.03025024
```

```
print("Manual Block Bootstrap Standard Errors (l=10):")
```

```
## [1] "Manual Block Bootstrap Standard Errors (l=10):"
```

```
print(manual_block_SE_10)
```

```
## [1] 0.46344448 0.46240722 0.03174143
```

```
print("Manual Block Bootstrap Standard Errors (l=20):")
```

```
## [1] "Manual Block Bootstrap Standard Errors (l=20):"
```

```
print(manual_block_SE_20)
```

```
## [1] 0.4471293 0.4454062 0.0317944
```

Comparison of “Manual” vs. **boot** Library Bootstrap Results

Below we compare the **manually coded** bootstrap methods with the results obtained through the **boot** package. Both approaches use the same ARMA(1,1) data and the same logic (residual resampling vs. block resampling), but they’re implemented differently.

MLE Baseline Estimates

Parameter	boot-Library Estimate	Manual Estimate (near-identical)
AR(1)	0.4726	~0.4726
MA(1)	-0.4176	~-0.4176
Intercept	0.0251	~0.0251

> **Note:** As expected, the fitted AR(1) and MA(1) parameters and intercept match closely because both methods call `arima(...)` on the same simulated series.

Residual Bootstrap SEs

boot-Library

- **AR(1)** SE = 0.6094
- **MA(1)** SE = 0.6119
- **Intercept** SE = 0.0287

Manual

- **AR(1)** SE ≈ 0.62
- **MA(1)** SE ≈ 0.62
- **Intercept** SE ≈ 0.028

Key Observations
1. **Highest Standard Errors:** Both methods confirm that the naive residual bootstrap overestimates uncertainty when ignoring dependence.
2. **Very Similar Results:** The manual-coded approach and the boot package yield nearly identical standard

errors (slight random differences due to sampling).
3. **Consistency:** This reaffirms that resampling residuals independently is not ideal for time series, as it does not preserve the autocorrelation structure.

Block Bootstrap SEs

boot-Library
- **l=5:** AR(1)=0.5175, MA(1)=0.5181, Intercept=0.0304
- **l=10:** AR(1)=0.4876, MA(1)=0.4854, Intercept=0.0301
- **l=20:** AR(1)=0.4820, MA(1)=0.4809, Intercept=0.0313

Manual
- **l=5:** AR(1)≈0.50, MA(1)≈0.50, Intercept≈0.030
- **l=10:** AR(1)≈0.46, MA(1)≈0.46, Intercept≈0.032
- **l=20:** AR(1)≈0.45, MA(1)≈0.45, Intercept≈0.032

Key Observations
1. **Consistently Lower SEs vs. Residual Bootstrap:** Both manual and boot-library block methods reduce the SEs substantially compared to the naive residual approach.
2. **Dependence Preservation:** Sampling in blocks helps capture the time-series correlation, leading to more realistic (usually smaller) standard errors.
3. **Similar Patterns in Block Size:** As `l` goes from 5 → 10 → 20, the AR/MA standard errors decrease slightly and "stabilize." Both manual and library methods show that moderate blocks (l=10 or 20) give consistent results.
4. **Close Agreement:** The manual-coded approach yields block-SE values that are numerically very close to the boot-library's. Minor differences arise from randomization and small differences in how blocks are sampled or how parameters converge in `arima()`.

2.3. Overall Conclusions and Recommendations

- **Residual Bootstrap** consistently shows the **largest** standard errors, both manually and via the library, confirming it can **overestimate** uncertainty by ignoring time dependence.
- **Block Bootstrap** standard errors are lower and more **reliable** for ARMA(1,1). Larger blocks (like 10 or 20) **stabilize** the estimates best.
- **Manual vs. Library**: In both methods, the **patterns and conclusions match**. The numerical differences are minimal and are purely random/fluctuations due to the bootstrap sampling itself.
- **Practical Advice**: For an ARMA(1,1) series, block sizes near 10–20 generally balance capturing correlation without oversmoothing. The block bootstrap is **strongly preferred** to a naive residual resampling if the goal is accurate inference on ARMA parameters.