HPI Simulator 5.0

The current production version of the HPI Simulator (4.4) generates simulated home price paths with some undesirable statistical properties:

1. The simulations bifurcate into two separate "high" and "low" distributions of HPI levels. This bifurcation becomes more pronounced when moving from initial to late simulation periods.
2. Price volatility decreases as simulated HPI paths move away farther away from the mean path specified in the RiskModel. Simulated price volatility should be independent of HPI levels.
3. The average volatilities of the simulated HPI paths do not match the values specified in the RiskModel.

The version 5.0 modifications to the HPI simulator are designed to fix these statistical problems.

(1) Input Data and Parameter Overview

RiskModel users control the paths of the HPI simulations by either specifying a constant annualized rate of home price appreciation (HPA) or by providing mean paths of HPI levels for each market. If a mean path is not provided for a market, the specified annual HPA rate generates a constant-slope mean path in log HPI levels. RiskModel users also specify annualized home price volatilities for each market to control the dispersion of simulated HPI paths.

The RiskModel calculates default values for the constant annualized HPA and volatilities based on the historical HPI for each market if users do not want to provide their own values. When these default values are used, the HPI simulations will follow paths that have average HPA rates and volatilities that match observed historical values. These default values are recalculated whenever updated HPI data is loaded into the RiskModel.

The HPI simulator also accounts for HPA correlations across different markets. For simulation purposes, these cross-market correlations are stored in static Cholesky decompositions of historical HPA correlation matrices -- one for the U.S. and states and another for metro areas. These Cholesky decomposition matrices were tabulated as part of the HPI simulator 5.0 calibration process and are not updated inside of the RiskModel.

Finally, three model simulation paramters control the HPI simulation paths for each market given the mean HPI paths (or constant annualized HPA rates) and annualized price volatilities specified by a RiskModel user (or their default values). The first two model parameters, which measure rates of HPA momentum and mean reversion, were estimated in the HPI simulator 5.0 calibration process and are not updated inside of the RiskModel. The third parameter is the residaul volatility in each market after removing momentum and mean reversion price volaility. The residual volatility paramters are updated in the RiskModel to reflect the user-specified (or default) annualized volatities for each market.

(2) Updating the Default Annualized HPA and Volatility Parameters

Historical HPI values are used to calculate the default annualized HPA and volatility parameters. However, before these values are calculated, the HPI must be seasonally-adjusted to remove the effects of seasonal home price flucations on the paramter estimates.

The seasonally-adjusted HPI estimates are calculated by applying a series of centered, non-truncated moving average filters to the HPI values.

$$ma_t(x, n) = \left(\sum_{max(t_0, t-s)}^{min(t+s, T)} x_t \right) / \left(min(t+s, T) - max(t_0, t-s) + 1\right)$$

where x is an unfiltered series, n is the (odd) moving average sample size, $s = (n-1)/2$, and t_0 and T are the start and end periods of the unfiltered series.

In [251]:
```
# R function for centered moving average

ma <- function(x, n) {

    t0 <- 1
    T <- length(x)
    x.ma <- rep(as.numeric(NA), T)
    s <- (n-1)/2

    for (t in t0:T) {
        lower.index <- max(t0, t-s)
        upper.index <- min(t+s, T)
        x.ma[t] <- sum(x[lower.index:upper.index])/(upper.index-lower.index+1)
    }

    return(x.ma)
}
```

The first step in the seasonal adjustment procedure is to transform the HPI to natural logs: $LnHPI_i = ln(HPI_i)$ for market i. Then calculate a 25-month, centered moving average of the log HPI values: $ma(LnHPI_i, 25)$. The differences between the unadjusted log HPI values and the 25-month moving average are the seasonal fluctuations in the log HPI values.

The next step is to apply two moving average filters (1st: $n = 5$, 2nd: $n = 3$) sequentially to the seasonal fluctuations. These calculations, however, are done separately for each calendar month. These moving averages are 'seasonal factors' -- the average seasonal fluctuations for particular calendar months averaged over multiple years.

The seasonally-adjusted log HPI series is equal to the differences between the unajusted series and the seasonal factors:

$$AdjLnHPI_i = LnHPI_i - SeasFac_i$$

In [252]:
```
# Load state-level HPI from a bulk export file

library(sqldf)
library(stringr)

hpi.state <- read.csv.sql('HPI_Bulk_Export_by_STATE_201808.csv',
                          sql = 'select STATE_CODE as state,
                                        substr(YYYYMM,5,2) as mm,
                                        YYYYMM as yyyymm,
                                        HOME_PRICE_INDEX as hpi
                                   from file
                                  where TIER_CODE = 11
                                  order by STATE_CODE, YYYYMM')
if (!is.null(getOption("sqldf.connection"))) sqldf()

# Pad state FIPS codes with leading zeros
hpi.state$state <- str_pad(as.character(hpi.state$state), 2, side=c("left"),
"0")

# Log transform HPI (in R, log = natural log)
hpi.state$ln.hpi <- log(hpi.state$hpi)

# Create list of states
states <- as.list(unique(hpi.state$state))

# Add columns for storing seasonal adjustment results
hpi.state[,c('seas.fluc','seas.fac','adj.ln.hpi')] <- as.numeric(NA)

# Loop over states
for (i in states) {

    # Extract rows for state
    hpi <- hpi.state[hpi.state$state==i,]

    # Create list of month labels
    mths <- as.list(unique(hpi$mm))

    # Calculate seasonal fluctuations
    ma25.ln.hpi <- ma(hpi$ln.hpi, 25)
    hpi$seas.fluc <- hpi$ln.hpi - ma25.ln.hpi

    # Calculate seasonal factors
    # Loop over calendar months
    for (j in mths) {

        # Extract hpi dataframe rows for given calendar month
        hpi.mth <- hpi[hpi$mm==j,]

        # 5-period (5-year) moving average
        ma5.seas.fluc <- ma(hpi.mth$seas.fluc, 5)

        # 3-period (3-year) moving average of 5-period (5-year) moving average
        ma3.ma5.seas.fluc <- ma(ma5.seas.fluc, 3)

        # Put seasonal factors into hpi dataframe
        hpi$seas.fac[hpi$mm==j] <- ma3.ma5.seas.fluc
```

```
    }

    # Calculate seasonally-adjusted log HPI
    hpi$adj.ln.hpi <- hpi$ln.hpi - hpi$seas.fac

    # Put results into hpi.state
    hpi.state$seas.fluc[hpi.state$state==i] <- hpi$seas.fluc
    hpi.state$seas.fac[hpi.state$state==i] <- hpi$seas.fac
    hpi.state$adj.ln.hpi[hpi.state$state==i] <- hpi$adj.ln.hpi

}

# Look at results for U.S.
head(hpi.state[hpi.state=='00',], n=24)
```

state	mm	yyyymm	hpi	ln.hpi	seas.fluc	seas.fac	adj.ln.hpi
00	01	197601	24.3566	3.192803	0.005546915	-0.005796533	3.198599
00	02	197602	23.4147	3.153364	-0.038769857	-0.016117760	3.169482
00	03	197603	23.1479	3.141904	-0.055811120	-0.017072398	3.158976
00	04	197604	23.0528	3.137787	-0.065835146	-0.018261925	3.156049
00	05	197605	23.2636	3.146890	-0.062757700	-0.016191108	3.163081
00	06	197606	24.0296	3.179286	-0.036949244	-0.004901896	3.184188
00	07	197607	24.1766	3.185385	-0.037299529	-0.004574902	3.189960
00	08	197608	24.3934	3.194313	-0.034782564	-0.003808455	3.198121
00	09	197609	24.4953	3.198481	-0.036678969	-0.007079532	3.205561
00	10	197610	24.8213	3.211702	-0.029338517	-0.007304138	3.219006
00	11	197611	25.0180	3.219596	-0.027111462	-0.009354193	3.228950
00	12	197612	25.3606	3.233197	-0.019037617	-0.009148447	3.242345
00	01	197701	25.5240	3.239619	-0.018333284	-0.004570926	3.244190
00	02	197702	25.9338	3.255547	-0.011058121	-0.013797444	3.269345
00	03	197703	26.4658	3.275853	-0.001599594	-0.014740979	3.290594
00	04	197704	26.9028	3.292230	0.002969409	-0.015699523	3.307930
00	05	197705	27.2772	3.306051	0.004313009	-0.014074223	3.320125
00	06	197706	27.8890	3.328232	0.013838447	-0.004004390	3.332237
00	07	197707	28.1844	3.338769	0.012450483	-0.003362490	3.342131
00	08	197708	28.5282	3.350893	0.012396611	-0.002599281	3.353492
00	09	197709	28.6875	3.356461	0.005777596	-0.005401097	3.361863
00	10	197710	28.9199	3.364530	0.001522662	-0.005532248	3.370062
00	11	197711	29.1183	3.371367	-0.003781120	-0.007400378	3.378767
00	12	197712	29.3521	3.379364	-0.007973562	-0.007114083	3.386478

The average annual HPA parameter value is calculated by converting the cumulative return for the entire seasonally-adjusted log HPI series to an annual frequency:

$$AvgHPA_i = (AdjLnHPI_{i,T} - AdjLnHPI_{i,t_0}) \times (12/(T-1))$$

The average annual price volatility paramter value is equal to the standard deviation of the 1st differences in the seasonally-adjusted log HPI series converted to an annual frequency:

$$AvgVol_i = StdDev\left(\Delta AdjLnHPI_i\right) \times \sqrt{12}$$

In [253]:
```
t0 <- 1
T <- nrow(hpi)

# Calculate average annual HPA and volatility for a subset of states
states <- list("00",   # National
               "04",   # Arizona
               "06")   # California

# Data frame for storing results
hist.values <- data.frame(avg.hpa=rep(as.numeric(NA),length(states)),
                          avg.vol=rep(as.numeric(NA),length(states)))
rownames(hist.values) <- unlist(states)

for (i in states) {

    cat(paste("State:",i,"\n"))

    # Extract rows for state
    hpi <- hpi.state[hpi.state$state==i,]

    # Average annual HPA
    avg.hpa <- (hpi$adj.ln.hpi[T] - hpi$adj.ln.hpi[t0])*(12/(T-1))
    cat(paste("Average Annual HPA:",avg.hpa,"\n"))

    # Average annual volatility
    avg.vol <- sd(diff(hpi$adj.ln.hpi, lag=1))*sqrt(12)
    cat(paste("Average Annual Volatility:",avg.vol,"\n"))

    # Store results
    hist.values[i,'avg.hpa'] = avg.hpa
    hist.values[i,'avg.vol'] = avg.vol

    cat("\n")

}

hist.values
```

```
State: 00
Average Annual HPA: 0.0494590039079743
Average Annual Volatility: 0.0209055663660761

State: 04
Average Annual HPA: 0.0442868920271275
Average Annual Volatility: 0.0294797594134768

State: 06
Average Annual HPA: 0.0635843098505731
Average Annual Volatility: 0.030910716750601
```

	avg.hpa	avg.vol
00	0.04945900	0.02090557
04	0.04428689	0.02947976
06	0.06358431	0.03091072

(3) Calculating the Value of the Residual Volatility Parameter

For the HPI Simulator 5.0, the price volatility for each market is assumed to be the sum of the volatility that can be predicted by the simulation model and the remaining unpredictable, residual volatility. The 5.0 simulation model is defined as:

$$Dev\Delta AdjLnHPI_{i,t}^{k} = \alpha_i \cdot ln(AdjLnHPI_{i,t-1}^{k} - \overline{AdjLnHPI}_{i,t-1}) + \beta_i(Dev\Delta AdjLnHPI_{i,t-1}^{k}) + \sum_{j}$$

where

$Dev\Delta AdjLnHPI_{i,t}^{k} = (\Delta AdjLnHPI_{i,t}^{k} - \overline{\Delta AdjLnHPI}_{i,t})$ is the home price change deviation from the mean HPA path for period t, market i, and simulation path k,

$\overline{AdjLnHPI}_{i,t-1}$ is the mean price level across all simulation paths,

α_i is a mean reversion coefficient for market i,

β_i is a momentum coefficient for market i,

C_{ij} is the Cholesky decompostion of the cross-market correlations of $Dev\Delta AdjLnHPI_i$,

σ_i is the monthly volatility of $Dev\Delta AdjLnHPI_i$ for market i, and

z_i is a standard normal disturbance term.

There are two sources of within-market volatility in the simulation model: (1) the volatility that is predicted by the mean reversion and momentum terms and (2) the unpredictable residual volatility, which is represented by the σ_i coefficient. (We ignore the cross-market correlations of $Dev\Delta AdjLnHPI_i$ when estimating the residual volatilities.)

The simulation model is essentially an AR(1) time series model, where β_i is the auto-correlation coefficient for $Dev\Delta AdjLnHPI_i$. (The estimated mean reversion coefficients, $\hat{\alpha}_i$, are very small compared to the estimated $\hat{\beta}_i$ coefficients, so it is O.K. to ignore them when calculating the residual volatilities of the simulation model.) For AR(1) formulation of the simulation model, the estimated residual variance is defined as:

$$\hat{\sigma}_i^2 = Var(Dev\Delta AdjLnHPI_i) \cdot (1 - \hat{\beta}_i^2)$$

When calculating the historical HPA and volatility paramters, it is assumed that HPA is constant over time (i.e., $\overline{\Delta AdjLnHPI}_{i,t}$ is a constant), so

$$\hat{\sigma}_i^2 = Var(\Delta AdjLnHPI_i) \cdot (1 - \hat{\beta}_i^2)$$

The residual volatility is equal to the square root of the residual variance.

Regression estimates of AR(1) coefficients are biased, however, so the estimates of $\hat{\sigma}_i$ will also be biased. To remove this bias, correction factors were calculated based on the observed annual HPA volatilites of the state- and CBSA-level HPI. For markets where $\hat{\beta}_i > 0.6$ the bias-corrected residual volality is equal to:

$$\hat{\sigma} = \sqrt{\hat{\sigma}_i^2 + exp(-11.9198 + 4.8306\hat{\beta}_i)/\sqrt{12}}$$

For markets where $\hat{\beta}_i \leq 0.6$ the bias-corrected residual volatiliy is equal to:

$$\hat{\sigma} = \sqrt{\hat{\sigma}_i^2 + exp(-10.83362 + 3.95718\hat{\beta}_i)}/\sqrt{12}$$

For a few CBSA-level markets, insufficient HPI data prevents the estimation of reliable simulation model coefficients. For these markets, the values of $\hat{\alpha}_i$ and $\hat{\beta}_i$ are set equal to zero and the residual volatility is set equal to the standard deviation to the historical HPA.

$$\hat{\sigma} = StdDev(\Delta AdjLnHPI_i^k)$$

In [254]:
```
# Calculate residual volatility for a subset of states
states <- list("00",   # National
               "04",   # Arizona
               "06")   # California

# Load state-level simulation model coefficients
sim.mod.coef <- read.csv("lm.mod.coef.states.csv",
                        colClasses=c("character","character","numeric","numeri
c"), row.names=c(1))
sim.mod.coef <- sim.mod.coef[unlist(states),]

# Add coloumn to hist.values
hist.values[,'resid.vol'] <- as.numeric(NA)

for (i in states) {

    # Convert annual volatility to monthly
    mth.vol <- hist.values[i,"avg.vol"]/sqrt(12)

    momentum <- sim.mod.coef[i,'momentum']

    # Calculate residual volatility estimates
    if (momentum!=0.0) {

        if (momentum>0.6) {

            resid.var <- (mth.vol^2) * (1 - (momentum)^2)
            resid.vol = sqrt(resid.var) + exp(-11.9198 + 4.8306*momentum)/sqrt
(12)

        } else {

            resid.var <- (mth.vol^2) * (1 - (momentum)^2)
            resid.vol = sqrt(resid.var) + exp(-10.83362 + 3.95718*momentum)/sq
rt(12)

        }

    } else {

        resid.vol = mth.vol

    }

    cat(paste("State:",i,"\n"))

    # Average annual volatility
    cat(paste("Average Annual Volatility:",hist.values[i,"avg.vol"],"\n"))

    # Average annual residual volatility
    cat(paste("Average Annual Residual Volatility:",resid.vol*sqrt(12),"\n"))

    # Store results
    hist.values[i,'resid.vol'] <- resid.vol*sqrt(12)

    cat("\n")
```

```
}

hist.values
```

```
State: 00
Average Annual Volatility: 0.0209055663660761
Average Annual Residual Volatility: 0.0117576621700212

State: 04
Average Annual Volatility: 0.0294797594134768
Average Annual Residual Volatility: 0.0107457419206369

State: 06
Average Annual Volatility: 0.030910716750601
Average Annual Residual Volatility: 0.0127070651574846
```

	avg.hpa	avg.vol	resid.vol
00	0.04945900	0.02090557	0.01175766
04	0.04428689	0.02947976	0.01074574
06	0.06358431	0.03091072	0.01270707

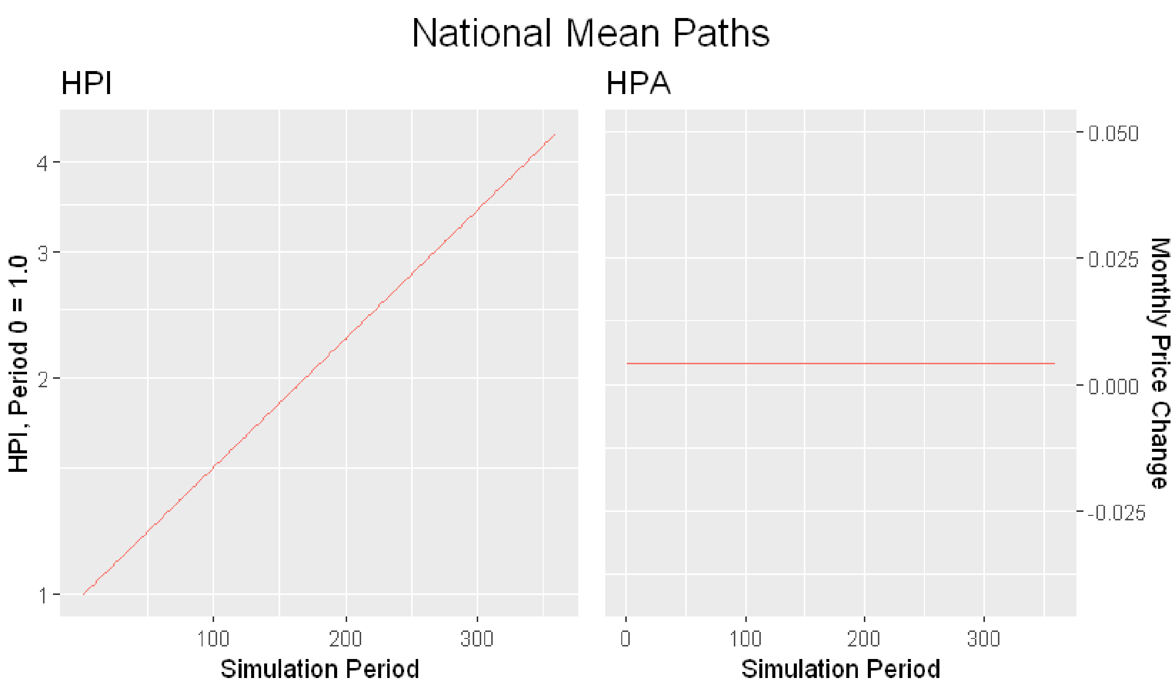
(4) Simulating HPI Paths (Default Paramter Values)

In [255]:
```
simulate.mod.multi.chol <- function(num.months, mean.hpa.path, initial.hpi, me
an.hpi.path,
                                    mod.coef, chol.mat) {

  # mean.hpa.path is the mean HPA path
  # mean.hpi.path is the mean HPI path

  # mod.coef are the simulation model coefficients
  # chol.mat is the Cholesky decomposition of the cross-market HPA correlation
 matrix

  # num.months is the number of simulation periods
  # num.markets is the number of markets in the simulation
  num.markets <- nrow(initial.hpi)

  # Create matrix for storing simulation results
  hpi <- matrix(NA, nrow=num.markets, ncol=num.months)
  rownames(hpi) <- rownames(initial.hpi)

  # Matrix of standard normal disturbances
  z <- matrix(rnorm(num.markets*num.months, 0, 1),
              nrow=num.months, ncol=num.markets)

  # Multiply by Cholesky matrix to get correlated disturbances
  z <- z %*% chol.mat

  # Multiply (element-by-element) by residual volatilities
  innov <- t(t(z) * mod.coef$sigma)

  # Loop across markets
  for (k in 1:num.markets) {

    market <- rownames(initial.hpi)[k]

    ch <- rep(NA, num.months)
    ch[1] <- 0

    hpi[market,1] <- log(initial.hpi[market,])

    diff.trend <- 0

    # Loop across simulation months
    for (i in 2:num.months) {

      # Simulate HPA deviation
      ch[i] <- mod.coef[market,"momentum"]*ch[i-1] +
        mod.coef[market,"reversion"]*diff.trend +
        innov[i,k]

      # Update simulated HPI level
      hpi[market,i] <- hpi[market,i-1] + ch[i] + mean.hpa.path[market,i]

      # Update HPI difference from mean HPI path
      diff.trend <- hpi[market,i] - mean.hpi.path[market,i]

    }
```

```
  }

  return(hpi)

}
```

In [256]:
```
# Load some helper functions for displaying results
source("(0)_Functions_Jupyter_Notebook.R")

generate.trend <- function(num.months, mean.hpa, initial.hpi) {
  if (length(mean.hpa)==1) {
    trend <- c(log(initial.hpi), log(initial.hpi)+cumsum(rep(mean.hpa, num.mon
ths-1)))
  } else {
    trend <- c(log(initial.hpi), log(initial.hpi)+cumsum(mean.hpa[2:length(mea
n.hpa)]))
  }
  return(trend)
}

# Run HPI simulation for a subset of states
states <- c("00",   # National
            "04",   # Arizona
            "06")   # California

num.markets <- 3    # Number of markets being simulated
num.months <- 360   # 30-year simulation horizon
num.sims <- 2000    # Number of simulation paths

# Extract HPI for subset of state
hpi <- hpi.state[hpi.state$state %in% states,]

# Set constant mean HPA rates to historical (monthly) averages
mean.hpa <- matrix(hist.values$avg.hpa/12, nrow=num.markets, ncol=1)
rownames(mean.hpa) <- states

# Convert mean HPA rate to mean HPA paths
mean.hpa.path <- t(tcrossprod(rep(1,num.months), mean.hpa))
rownames(mean.hpa.path) <- states

# Set initial HPI values for simulation
initial.hpi <- matrix(1.0, nrow=num.markets, ncol=1)
rownames(initial.hpi) <- states

# Generate mean HPI paths by applying mean HPA paths to initial HPI values
mean.hpi.path <- matrix(NA, nrow=num.markets, ncol=num.months)
rownames(mean.hpi.path) <- states
for (i in 1:length(states)) {
  mean.hpi.path[i,] <- exp(c(log(initial.hpi[i,1]),
                             log(initial.hpi[i,1])+cumsum(mean.hpa.path[i,2:nu
m.months])))
}
dimnames(mean.hpi.path) <- list(market=rownames(initial.hpi),
                                period=c(1:dim(mean.hpi.path)[2]))
```

In [257]:
```
library(grid)
library(gridExtra)

# Plot national mean HPA and HPI paths
state <- "00"

path.df <- data.frame(period=1:num.months,
                      mean.hpa.path=mean.hpa.path[state,],
                      mean.hpi.path=mean.hpi.path[state,])

options(repr.plot.height=4)
plot1 <- ggplot(path.df, aes(x = period)) +
            geom_line(aes(y=mean.hpi.path, colour="Mean HPI Path")) +
            ggtitle("HPI") +
            coord_trans(y = "log") +
            xlab("Simulation Period") +
            ylab("HPI, Period 0 = 1.0") +
            theme(legend.position="none")
plot2 <- ggplot(path.df, aes(x = period)) +
            geom_line(aes(y=mean.hpa.path, colour="Mean HPA Path")) +
            ggtitle("HPA") +
            scale_y_continuous(position = "right") +
            xlab("Simulation Period") +
            ylab("Monthly Price Change") +
            theme(legend.position="none")
grid.arrange(plot1, plot2, ncol=2, top = textGrob("National Mean Paths",gp=gpa
r(fontsize=16)))
```

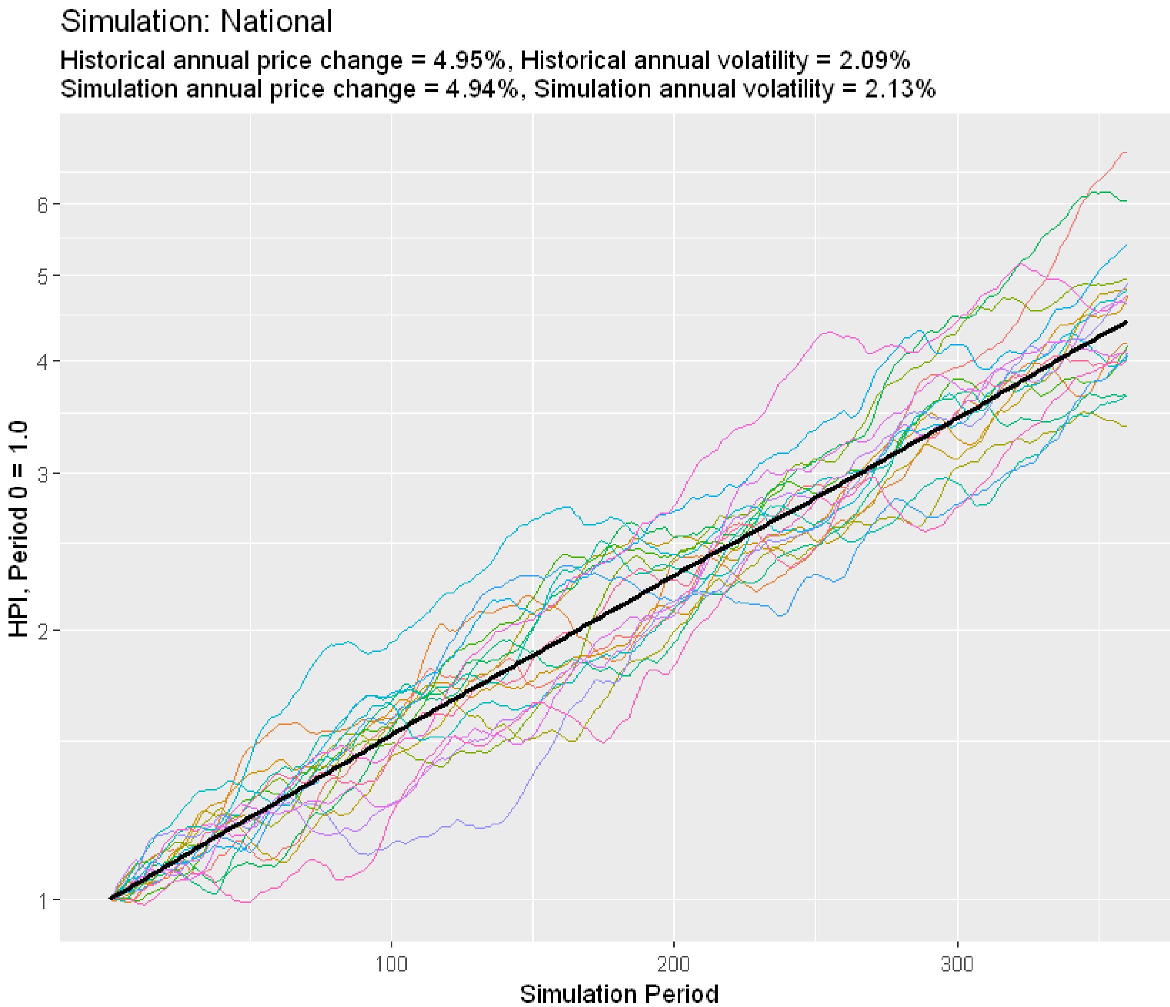


In [258]:
```
# Create array for storing HPI simulation results
    # Dimension 1: states
    # Dimension 2: simulation
    # Dimension 3: month
hpi.array <- array(NA, dim=c(nrow(initial.hpi), num.sims, num.months))
dimnames(hpi.array) <- list(market=rownames(initial.hpi),
                            sim=c(1:dim(hpi.array)[2]),
                            period=c(1:dim(hpi.array)[3]))

# Load HPA correlation matrix (only used for printing correlation targets)
cor.mat <- read.csv("cor.state.csv", header=TRUE, row.names=1)

state.codes <- str_pad(rownames(cor.mat), 2, side=c("left"), "0")
cor.mat <- as.matrix(cor.mat)
rownames(cor.mat) <- state.codes
colnames(cor.mat) <- state.codes
cor.mat <- cor.mat[states, states]   # Keep rows and columns for target states

# Load HPA correlation Cholesky decomposition matrix (used in simulation)
chol.mat <- read.csv("chol.state.csv", header=TRUE, row.names=1)

state.codes <- str_pad(rownames(chol.mat), 2, side=c("left"), "0")
chol.mat <- as.matrix(chol.mat)
rownames(chol.mat) <- state.codes
colnames(chol.mat) <- state.codes
chol.mat <- chol.mat[states, states]   # Keep rows and columns for target stat
es

# Load state-level simulation model coefficients
sim.mod.coef <- read.csv("lm.mod.coef.states.csv",
                        colClasses=c("character","character","numeric","numeri
c"), row.names=c(1))
sim.mod.coef <- sim.mod.coef[unlist(states),]

# Add sigma coefficients to model coefficients
#   Sigma = monthly residual volatlity
sim.mod.coef$sigma <- hist.values$resid.vol/sqrt(12)

# Run simulations
for (i in 1:num.sims) {

  hpi.array[,i,] <- exp(simulate.mod.multi.chol(num.months, mean.hpa.path, ini
tial.hpi,
                                                log(mean.hpi.path), sim.mod.co
ef, chol.mat))

}

r_multi <- summ.by.sim(hpi.array)


cat("\n")
cat("Annual Mean Returns:\n")
ret.mean <- aggregate((ret.mean*12)~market, data=r_multi, mean)
ret.mean[,2] <- format(ret.mean[,2], digits=5)
ret.mean[,3] <- format(mean.hpa*12, digits=5)
```

```
colnames(ret.mean) <- c('state','ret.mean','target')
ret.mean$state <- states
print(ret.mean, row.names=FALSE)

cat("\n")
cat("Annual Return Standard Deviations:\n")
sd.mean <- aggregate((ret.sd*sqrt(12))~market, data=r_multi, mean)
sd.mean[,2] <- format(sd.mean[,2], digits=5)
sd.mean[,3] <- format(hist.values$avg.vol[order(rownames(hist.values))], digit
s=5)
colnames(sd.mean) <- c('state','sd.mean','target')
sd.mean$state <- states
print(sd.mean, row.names=FALSE)

cat("\n")
cat("Target Residual Correlations:\n")
rho.df <- data.frame(cor.mat,
                     row.names=rownames(cor.mat))
colnames(rho.df) <- colnames(cor.mat)
print(format(rho.df, digits=3))

cat("\n")
cat("Simulated Residual Correlations:\n")
cor.mat <- hpa_dev.cor(hpi.array, mean.hpa)
print(format(cor.mat, digits=3))
```

```
Annual Mean Returns:
 state ret.mean   target
    00 0.049357 0.049459
    04 0.044087 0.044287
    06 0.063651 0.063584

Annual Return Standard Deviations:
 state  sd.mean   target
    00 0.021318 0.020906
    04 0.030184 0.029480
    06 0.031279 0.030911

Target Residual Correlations:
      00    04    06
00 1.000 0.188 0.283
04 0.188 1.000 0.127
06 0.283 0.127 1.000

Simulated Residual Correlations:
      00    04    06
00 1.000 0.163 0.267
04 0.163 1.000 0.127
06 0.267 0.127 1.000
```

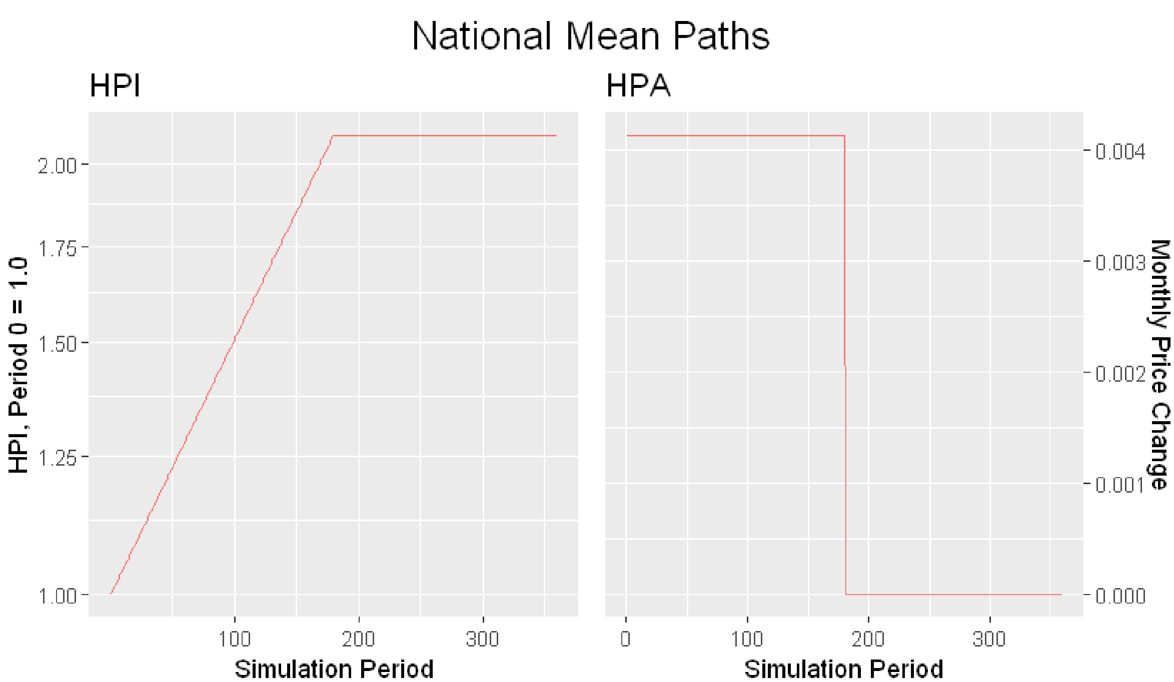
In [259]:
```
state <- "00"

hpi.array.us <- hpi.array[state,,]

hpi.paths <- as.data.frame(as.table(hpi.array.us))
hpi.paths$period <- as.integer(hpi.paths$period)
colnames(hpi.paths)[3] <- "value"

options(repr.plot.height=6)
print(ggplot(hpi.paths, aes(period, value)) +
        geom_line(aes(colour = sim), data=function(x){x[x$sim %in% as.characte
r(1:20), ]}) +
        stat_summary(aes(y = value), fun.y=mean, colour="black",
                     geom="line", size=1) +
        coord_trans(y = "log") +
        theme(legend.position="none") +
        ggtitle(paste("Simulation: National"),
                subtitle=(paste0("Historical annual price change = ",
                                 percent(as.numeric(mean.hpa[state,])*12),
                                 ", Historical annual volatility = ",
                                 percent(hist.values[state,'avg.vol']),
                                 "\n",
                                 "Simulation annual price change = ",
                                 percent(as.numeric(ret.mean[ret.mean$state==s
tate,"ret.mean"])),
                                 ", Simulation annual volatility = ",
                                 percent(as.numeric(sd.mean[sd.mean$state==sta
te,"sd.mean"]))))) +
        xlab("Simulation Period") +
        ylab("HPI, Period 0 = 1.0"))
```



(5) Simulate HPI Paths (User-Specified Mean HPA Paths)

In [260]:
```
# Run HPI simulation for a subset of states
states <- c("00",   # National
            "04",   # Arizona
            "06")   # California

num.markets <- 3    # Number of markets being simulated
num.months <- 360   # 30-year simulation horizon
num.sims <- 2000    # Number of simulation paths

# Extract HPI for subset of state
hpi <- hpi.state[hpi.state$state %in% states,]

# Set constant mean HPA rates to historical (monthly) averages
# But flatline HPA for last 15 years of simulation
mean.hpa <- matrix(hist.values$avg.hpa/12, nrow=num.markets, ncol=1)
rownames(mean.hpa) <- states

# Convert mean HPA rate to mean HPA paths
mean.hpa.path <- t(tcrossprod(rep(1,num.months), mean.hpa))
mean.hpa.path[,181:360] <- rep(0, num.markets)
rownames(mean.hpa.path) <- states

# Set initial HPI values for simulation
initial.hpi <- matrix(1.0, nrow=num.markets, ncol=1)
rownames(initial.hpi) <- states

# Generate mean HPI paths by applying mean HPA paths to initial HPI values
mean.hpi.path <- matrix(NA, nrow=num.markets, ncol=num.months)
rownames(mean.hpi.path) <- states
for (i in 1:length(states)) {
  mean.hpi.path[i,] <- exp(c(log(initial.hpi[i,1]),
                             log(initial.hpi[i,1])+cumsum(mean.hpa.path[i,2:nu
m.months])))
}
dimnames(mean.hpi.path) <- list(market=rownames(initial.hpi),
                                period=c(1:dim(mean.hpi.path)[2]))
```

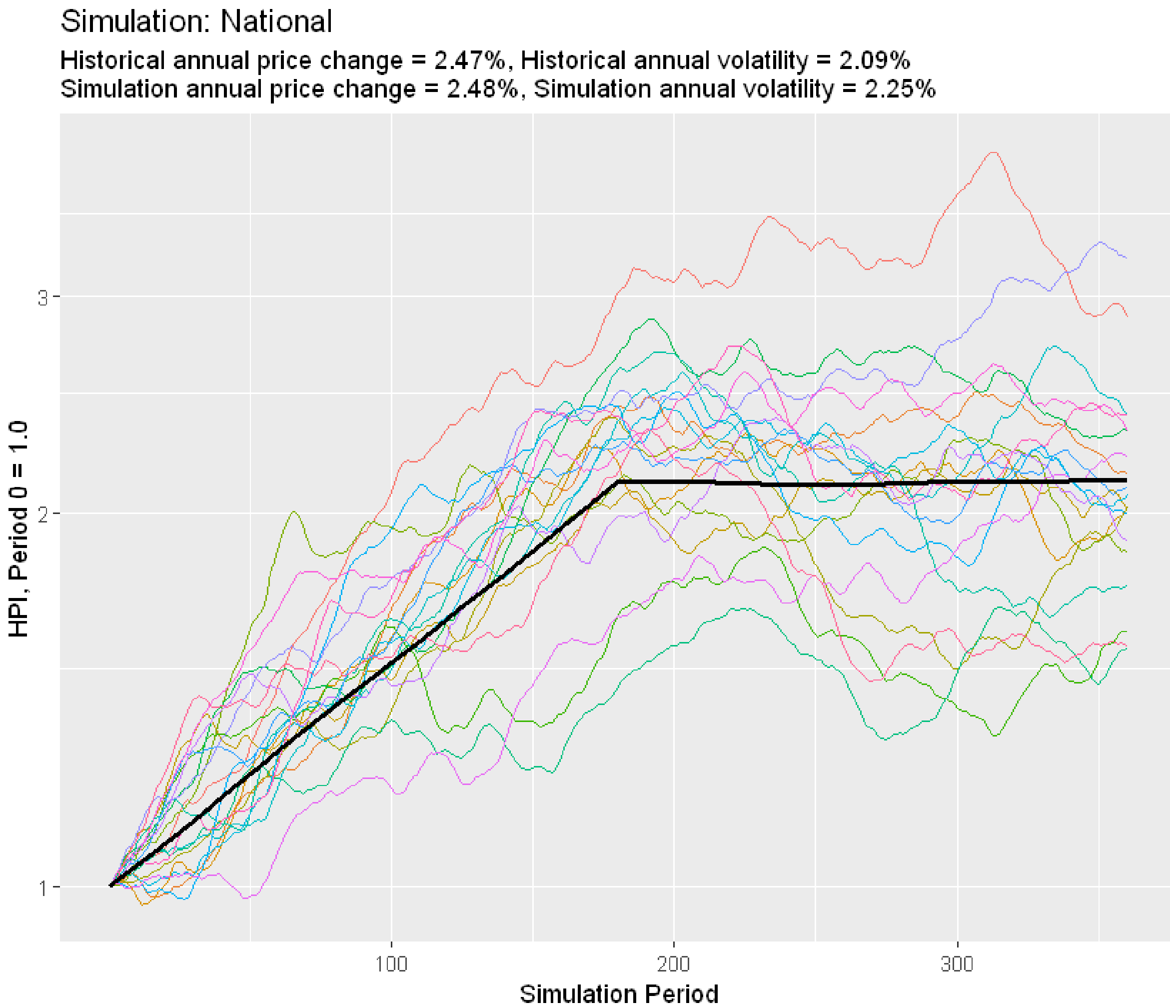
In [261]:
```
library(grid)
library(gridExtra)

# Plot national mean HPA and HPI paths
state <- "00"

path.df <- data.frame(period=1:num.months,
                      mean.hpa.path=mean.hpa.path[state,],
                      mean.hpi.path=mean.hpi.path[state,])

options(repr.plot.height=4)
plot1 <- ggplot(path.df, aes(x = period)) +
            geom_line(aes(y=mean.hpi.path, colour="Mean HPI Path")) +
            ggtitle("HPI") +
            coord_trans(y = "log") +
            xlab("Simulation Period") +
            ylab("HPI, Period 0 = 1.0") +
            theme(legend.position="none")
plot2 <- ggplot(path.df, aes(x = period)) +
            geom_line(aes(y=mean.hpa.path, colour="Mean HPA Path")) +
            ggtitle("HPA") +
            scale_y_continuous(position = "right") +
            xlab("Simulation Period") +
            ylab("Monthly Price Change") +
            theme(legend.position="none")
grid.arrange(plot1, plot2, ncol=2, top = textGrob("National Mean Paths",gp=gpa
r(fontsize=16)))
```



In [262]:
```
# Create array for storing HPI simulation results
    # Dimension 1: states
    # Dimension 2: simulation
    # Dimension 3: month
hpi.array <- array(NA, dim=c(nrow(initial.hpi), num.sims, num.months))
dimnames(hpi.array) <- list(market=rownames(initial.hpi),
                            sim=c(1:dim(hpi.array)[2]),
                            period=c(1:dim(hpi.array)[3]))


# Run simulations
for (i in 1:num.sims) {

  hpi.array[,i,] <- exp(simulate.mod.multi.chol(num.months, mean.hpa.path, ini
tial.hpi,
                                                log(mean.hpi.path), sim.mod.co
ef, chol.mat))

}

r_multi <- summ.by.sim(hpi.array)


cat("\n")
cat("Annual Mean Returns:\n")
ret.mean <- aggregate((ret.mean*12)~market, data=r_multi, mean)
ret.mean[,2] <- format(ret.mean[,2], digits=5)
ret.mean[,3] <- format((mean.hpa/2)*12, digits=5)
colnames(ret.mean) <- c('state','ret.mean','target')
ret.mean$state <- states
print(ret.mean, row.names=FALSE)

cat("\n")
cat("Annual Return Standard Deviations:\n")
sd.mean <- aggregate((ret.sd*sqrt(12))~market, data=r_multi, mean)
sd.mean[,2] <- format(sd.mean[,2], digits=5)
sd.mean[,3] <- format(hist.values$avg.vol[order(rownames(hist.values))], digit
s=5)
colnames(sd.mean) <- c('state','sd.mean','target')
sd.mean$state <- states
print(sd.mean, row.names=FALSE)

cat("\n")
cat("Target Residual Correlations:\n")
rho.df <- data.frame(cor.mat,
                     row.names=rownames(cor.mat))
colnames(rho.df) <- colnames(cor.mat)
print(format(rho.df, digits=3))

cat("\n")
cat("Simulated Residual Correlations:\n")
cor.mat <- hpa_dev.cor(hpi.array, mean.hpa)
print(format(cor.mat, digits=3))
```

```
Annual Mean Returns:
 state ret.mean   target
    00 0.024800 0.024730
    04 0.022283 0.022143
    06 0.031784 0.031792

Annual Return Standard Deviations:
 state  sd.mean   target
    00 0.022546 0.020906
    04 0.030808 0.029480
    06 0.032612 0.030911

Target Residual Correlations:
      00    04    06
00 1.000 0.163 0.267
04 0.163 1.000 0.127
06 0.267 0.127 1.000

Simulated Residual Correlations:
      00    04    06
00 1.000 0.217 0.329
04 0.217 1.000 0.171
06 0.329 0.171 1.000
```

In [263]:
```
state <- "00"

hpi.array.us <- hpi.array[state,,]

hpi.paths <- as.data.frame(as.table(hpi.array.us))
hpi.paths$period <- as.integer(hpi.paths$period)
colnames(hpi.paths)[3] <- "value"

options(repr.plot.height=6)
print(ggplot(hpi.paths, aes(period, value)) +
        geom_line(aes(colour = sim), data=function(x){x[x$sim %in% as.characte
r(1:20), ]}) +
        stat_summary(aes(y = value), fun.y=mean, colour="black",
                     geom="line", size=1) +
        coord_trans(y = "log") +
        theme(legend.position="none") +
        ggtitle(paste("Simulation: National"),
                subtitle=(paste0("Historical annual price change = ",
                                 percent(as.numeric(mean.hpa[state,]/2)*12),
                                 ", Historical annual volatility = ",
                                 percent(hist.values[state,'avg.vol']),
                                 "\n",
                                 "Simulation annual price change = ",
                                 percent(as.numeric(ret.mean[ret.mean$state==s
tate,"ret.mean"])),
                                 ", Simulation annual volatility = ",
                                 percent(as.numeric(sd.mean[sd.mean$state==sta
te,"sd.mean"]))))) +
        xlab("Simulation Period") +
        ylab("HPI, Period 0 = 1.0"))
```


Simulation: National
Historical annual price change = 2.47%, Historical annual volatility = 2.09%
Simulation annual price change = 2.48%, Simulation annual volatility = 2.25%
HPI, Period 0 = 1.0
3
2
1
100
200
300
Simulation Period