Chess Notes

James Otto

11/15/22

Table of contents

Preface

These are my notes on various topics in chess, aggregated from a variety of sources.

This is a Quarto book. To learn more about Quarto books visit https://quarto.org/docs/books.

Part I

Openings

The white pieces

1 The Scotch

```
board_setup = function(data, title) {

  var game = data.find(d => d.title === title)

  // if there's just one variation, need to put in array
  var variations = game.variations

  var pos = game.start - 1
  var moved = false

  var onDragMove = function() {
    moved = true

    d3.select('#reset-' + title)
      .classed('btn-outline-secondary', false)
      .classed('btn-success', true)

    d3.select('#analyze-' + title).property("disabled", true)
    d3.select('#prev-' + title).property("disabled", true)
    d3.select('#next-' + title).property("disabled", true)
  }

  // Note: boolean arrays don't survive transpose(), need to use characters ("show" and "h
  var board = Chessboard(title, {showNotation: game.notation === 'show', position: game.po

  // Fix touch controls on mobile
  jQuery('#' + title).on('scroll touchmove touchend touchstart contextmenu', function(e){
    e.preventDefault();
  });

  // flip initial orientation if needed
  if (game.orientation === "black") {
    board.flip()
  }
```

```
// resize board if window is resized
$(window).resize(board.resize)

d3.select('#prev-' + title).property("disabled", pos == 0)
d3.select('#next-' + title).property("disabled", pos == game.positions.length - 1)

$('#next-' + title).on('click', function () {
  if (pos < game.positions.length - 1) {
    pos = pos + 1
    board.position(game.positions[pos])
  }

  d3.select('#analyze-' + title).property("disabled", false)
  d3.select('#prev-' + title).property("disabled", false)
  d3.select('#next-' + title).property("disabled", pos == game.positions.length - 1)
})

$('#prev-' + title).on('click', function () {
  if (pos > 0) {
    pos = pos - 1
    board.position(game.positions[pos])
  }

  d3.select('#analyze-' + title).property("disabled", false)
  d3.select('#prev-' + title).property("disabled", pos == 0)
  d3.select('#next-' + title).property("disabled", false)
})

$('#reset-' + title).on('click', function() {

  // if pieces have not been moved, reset to beginning of line
  if (!moved) {

    // if looking at a variation, reset to main line
    if (!(variations === undefined) && variations.includes(game.title)) {

      d3.select('#' + game.title)
        .style("box-shadow", null)
        .style("color", null)

      game = data.find(d => d.title === title)
```

```
    pos = game.start - 1

    d3.select('#reset-' + title)
      .classed('btn-primary', false)
      .classed('btn-outline-secondary', true)

  // Otherwise, reset main line (possibly to start of game)
  } else {

    // if reset button is pressed at starting position, totally reset board
    if (pos === game.start - 1) {
      pos = 0
    } else {
      pos = game.start - 1
    }

  }

} else {

  // color reset button correctly if pieces are moved:
  if (!(variations === undefined) && variations.includes(game.title)) {

    d3.select('#reset-' + title)
      .classed('btn-primary', true)
      .classed('btn-success', false)

  } else {

    d3.select('#reset-' + title)
      .classed('btn-outline-secondary', true)
      .classed('btn-success', false)

  }
}

board.position(game.positions[pos])

d3.select('#analyze-' + title).property("disabled", false)
d3.select('#prev-' + title).property("disabled", pos == 0)
d3.select('#next-' + title).property("disabled", pos == game.positions.length - 1)
```

```
  moved = false

})

$('#analyze-' + title).on('click', function() {
  window.open(game.link[pos], '_blank');
})

$('#flip-' + title).on('click', board.flip)


if (!(variations === undefined)) {

  if (!(Array.isArray(variations))) {
    variations = [variations]
  }

  for (let k = 0; k < variations.length; k++) {
    $('#' + variations[k]).on('click', function() {

      // if clicked on active variation, reset to main line
      if (game.title === variations[k]) {

        d3.select('#' + game.title)
          .style("box-shadow", null)
          .style("color", null)

        game = data.find(d => d.title === title)

        // Don't reset position if before start of game
        if (pos >= game.start) {
          pos = game.start - 1
        }

        board.position(game.positions[pos])

        d3.select('#reset-' + title)
          .classed('btn-primary', false)
          .classed('btn-outline-secondary', true)

      // otherwise, set to selected variation
```

```
    } else {

      game = data.find(d => d.title === variations[k])

      // Don't reset position if before start
      // (this allows for the complete viewing of variations)
      if (pos >= game.start) {
        pos = game.start - 1
      }

      board.position(game.positions[pos])

      d3.select('#reset-' + title)
        .classed('btn-outline-secondary', false)
        .classed('btn-success', false)
        .classed('btn-primary', true)

    // need to unset styling for other variations
    variations.filter(name => !(name === variations[k]))
      .map(name =>  {
        d3.select('#' + name)
          .style("box-shadow", null)
          .style("color", null)
          //.style("box-shadow", "inset 0 0 0 0 #2780e3")
          //.style("color", "#2780e3")
      })

    // indicate variation is selected
    d3.select('#' + variations[k])
      .style("box-shadow", "inset 10vw 0 0 0 #2780e3")
      .style("color", "white")

    }

    // activate/deactivate UI
    d3.select('#analyze-' + title).property("disabled", false)
    d3.select('#prev-' + title).property("disabled", pos == 0)
    d3.select('#next-' + title).property("disabled", pos == game.positions.length - 1)

  })
}
```

```
  }

}
```

```
scotch = transpose(scotch_t)
```

```
scotch.filter(d => d.type === 'main')
  .map(d => board_setup(scotch, d.title))
```

1.1 Introduction

1.2 Main Line (4.Nxd4)

1.2.1 Classical Variation (4...Bc5)

- After 4.Nxd4 Bc5 there are three main lines:
  - 5.Be3 (the most popular response)
    - Note: 8...Bb6 is protecting the bishop
    - Instead of 7...O-O, 7...Ne5 is also possible
  - 5.Nb3 and 5.Nxc6

1.2.2 Schmidt Variation (4...Nf6)

- After 4.Nxd4 Nf6 there are two options:
  - The Mieses Variation: 5.Nxc6
  - The Scotch Four Knights Game: 5.Nc3 Bb4 or 5...Bc5

1.2.3 Lolli Variation (4...Nxd4?)

The response to 4...Nxd4 is 5.Qxd4, black cannot attack the queen. Any attempt results in a poor position:

- 5...Nf6 6.e5 threatens to trap the knight
  - 6...c5 is black's best option (still bad)
  - 6...Ng8 also saves the knight at a severe cost

- 5…c5 6.Qd3 d5 loses the d pawn
  - 6…d6 postpones the loss, but after O-O-O and e4-e5 it will fall
- 5…Qf6 leads to a poor position
  - 8…f6 loses a pawn
  - 9…Bxd6 also loses a pawn

1.2.4 Other fourth moves for black

- Black has a few other options, besides 4…Bc5 and 4…Nf6:
  - 4…Qf6 typically transposes into the mainline
  - 4…Qh4 (the dubious Steinitz Variation)
  - 4…Bb4 again, black has to retreat 6…Bb6

1.3 Scotch Gambit (4.Bc4)

- Black can accept with 4…Bc5 5.c3 dxc3 .
  - Alternatively, 5…Nf6 transitions into the Italian Game
- 4…Bb4+ is the London Defense

1.4 Traps

1.4.1 Classical Variation (4.Nxd4 Bc5)

- If black doesn't defend the bishop on c5 after 4.Nxd5 Bc5 5.Be3, white can take the knight on c6, winning the bishop!
  - For example, 5….Nf6 loses the bishop
  - 5….d6 defends the bishop, but it's not pretty

1.4.2 3…d6

- 3…d6 4.dxe5 dxe5 5. Qxd8 is bad for black
  - If 5…Kxd8 , black loses the ability to castle and possibly the rook on h8
  - But, 5…Nxd8 loses a pawn

References

Download PGN

2 Petrov's Defense

```
board_setup = function(data, title) {

  var game = data.find(d => d.title === title)

  // if there's just one variation, need to put in array
  var variations = game.variations

  var pos = game.start - 1
  var moved = false

  var onDragMove = function() {
    moved = true

    d3.select('#reset-' + title)
      .classed('btn-outline-secondary', false)
      .classed('btn-success', true)

    d3.select('#analyze-' + title).property("disabled", true)
    d3.select('#prev-' + title).property("disabled", true)
    d3.select('#next-' + title).property("disabled", true)
  }

  // Note: boolean arrays don't survive transpose(), need to use characters ("show" and "h
  var board = Chessboard(title, {showNotation: game.notation === 'show', position: game.po

  // Fix touch controls on mobile
  jQuery('#' + title).on('scroll touchmove touchend touchstart contextmenu', function(e){
    e.preventDefault();
  });

  // flip initial orientation if needed
  if (game.orientation === "black") {
    board.flip()
  }
```

```
// resize board if window is resized
$(window).resize(board.resize)

d3.select('#prev-' + title).property("disabled", pos == 0)
d3.select('#next-' + title).property("disabled", pos == game.positions.length - 1)

$('#next-' + title).on('click', function () {
  if (pos < game.positions.length - 1) {
    pos = pos + 1
    board.position(game.positions[pos])
  }

  d3.select('#analyze-' + title).property("disabled", false)
  d3.select('#prev-' + title).property("disabled", false)
  d3.select('#next-' + title).property("disabled", pos == game.positions.length - 1)
})

$('#prev-' + title).on('click', function () {
  if (pos > 0) {
    pos = pos - 1
    board.position(game.positions[pos])
  }

  d3.select('#analyze-' + title).property("disabled", false)
  d3.select('#prev-' + title).property("disabled", pos == 0)
  d3.select('#next-' + title).property("disabled", false)
})

$('#reset-' + title).on('click', function() {

  // if pieces have not been moved, reset to beginning of line
  if (!moved) {

    // if looking at a variation, reset to main line
    if (!(variations === undefined) && variations.includes(game.title)) {

      d3.select('#' + game.title)
        .style("box-shadow", null)
        .style("color", null)

      game = data.find(d => d.title === title)
```

```
      pos = game.start - 1

      d3.select('#reset-' + title)
        .classed('btn-primary', false)
        .classed('btn-outline-secondary', true)

    // Otherwise, reset main line (possibly to start of game)
    } else {

      // if reset button is pressed at starting position, totally reset board
      if (pos === game.start - 1) {
        pos = 0
      } else {
        pos = game.start - 1
      }

    }

  } else {

    // color reset button correctly if pieces are moved:
    if (!(variations === undefined) && variations.includes(game.title)) {

      d3.select('#reset-' + title)
        .classed('btn-primary', true)
        .classed('btn-success', false)

    } else {

      d3.select('#reset-' + title)
        .classed('btn-outline-secondary', true)
        .classed('btn-success', false)

    }
  }

  board.position(game.positions[pos])

  d3.select('#analyze-' + title).property("disabled", false)
  d3.select('#prev-' + title).property("disabled", pos == 0)
  d3.select('#next-' + title).property("disabled", pos == game.positions.length - 1)
```

```
  moved = false

})

$('#analyze-' + title).on('click', function() {
  window.open(game.link[pos], '_blank');
})

$('#flip-' + title).on('click', board.flip)


if (!(variations === undefined)) {

  if (!(Array.isArray(variations))) {
    variations = [variations]
  }

  for (let k = 0; k < variations.length; k++) {
    $('#' + variations[k]).on('click', function() {

      // if clicked on active variation, reset to main line
      if (game.title === variations[k]) {

        d3.select('#' + game.title)
          .style("box-shadow", null)
          .style("color", null)

        game = data.find(d => d.title === title)

        // Don't reset position if before start of game
        if (pos >= game.start) {
          pos = game.start - 1
        }

        board.position(game.positions[pos])

        d3.select('#reset-' + title)
          .classed('btn-primary', false)
          .classed('btn-outline-secondary', true)

      // otherwise, set to selected variation
```

```
    } else {

      game = data.find(d => d.title === variations[k])

      // Don't reset position if before start
      // (this allows for the complete viewing of variations)
      if (pos >= game.start) {
        pos = game.start - 1
      }

      board.position(game.positions[pos])

      d3.select('#reset-' + title)
        .classed('btn-outline-secondary', false)
        .classed('btn-success', false)
        .classed('btn-primary', true)

    // need to unset styling for other variations
    variations.filter(name => !(name === variations[k]))
      .map(name =>  {
        d3.select('#' + name)
          .style("box-shadow", null)
          .style("color", null)
          //.style("box-shadow", "inset 0 0 0 0 #2780e3")
          //.style("color", "#2780e3")
      })

    // indicate variation is selected
    d3.select('#' + variations[k])
      .style("box-shadow", "inset 10vw 0 0 0 #2780e3")
      .style("color", "white")

    }

    // activate/deactivate UI
    d3.select('#analyze-' + title).property("disabled", false)
    d3.select('#prev-' + title).property("disabled", pos == 0)
    d3.select('#next-' + title).property("disabled", pos == game.positions.length - 1)

  })
}
```

```
  }

}
```

```
petrov = transpose(petrov_t)
```

```
petrov.filter(d => d.type === 'main')
  .map(d => board_setup(petrov, d.title))
```

2.1 Introduction

Petrov's Defense begins with 1.e4 e5 2.Nf3 Nf6. White has two main options for their third move: 3.Nxe5, the Classical Variation and 3.d4, the Steinitz Variation—here I focus on 3.Nxe5

2.2 Classical Variation (3.Nxe5 d6)

3…d6 is Black's best response to 3.Nxe5, followed with 4.Nf3 Nxe4

- White's typical response is 5.d4 , white is looking to kick Black's advanced knight from e4 with moves like c4 and Re1
- An alternative option for white is 5.Nc3 , looking for Be3, Qd2, and 0-0-0 with a kingside attack
  - If 5…Bf5 , Black loses a piece

2.3 The Damiano Defense (3.Nxe5 Nxe4)

- 3…Nxe4 is bad for Black, 4.Qe2 poses quite a problem.
  - 4…Nf6 loses the queen
  - 4…d5 loses a pawn
  - 4…Qe7 is "best"

2.4 Stafford Gambit (3.Nxe5 Nc6)

3…Nc6 is the dubious but tricky Stafford Gambit. Once refuted, White has the advantage. The key initial moves for white are 5.d3 (defending e4) followed by 6.Be2 (developing the bishop while defending against Ng4).

- At this point, Black has two options:
  - If 6…h5 , calmly kick the bishop
  - If 6…Ng4 , take the knight with 7.Bxg4 then defend with the queen
- In the latter, a queen trade is almost forced—Black may avoid this with 9…Qh5
  - This is tricky though, 11…Bxc2 and 12…Bxc2 are both bad ideas

2.5 Other responses to 3.Nxe5

- 3…Qe7 is mildly worse for Black

References

Download PGN

The black pieces

3 Scandinavian Defense

```
board_setup = function(data, title) {

  var game = data.find(d => d.title === title)

  // if there's just one variation, need to put in array
  var variations = game.variations

  var pos = game.start - 1
  var moved = false

  var onDragMove = function() {
    moved = true

    d3.select('#reset-' + title)
      .classed('btn-outline-secondary', false)
      .classed('btn-success', true)

    d3.select('#analyze-' + title).property("disabled", true)
    d3.select('#prev-' + title).property("disabled", true)
    d3.select('#next-' + title).property("disabled", true)
  }

  // Note: boolean arrays don't survive transpose(), need to use characters ("show" and "h
  var board = Chessboard(title, {showNotation: game.notation === 'show', position: game.po

  // Fix touch controls on mobile
  jQuery('#' + title).on('scroll touchmove touchend touchstart contextmenu', function(e){
    e.preventDefault();
  });

  // flip initial orientation if needed
  if (game.orientation === "black") {
    board.flip()
  }
```

```
// resize board if window is resized
$(window).resize(board.resize)

d3.select('#prev-' + title).property("disabled", pos == 0)
d3.select('#next-' + title).property("disabled", pos == game.positions.length - 1)

$('#next-' + title).on('click', function () {
  if (pos < game.positions.length - 1) {
    pos = pos + 1
    board.position(game.positions[pos])
  }

  d3.select('#analyze-' + title).property("disabled", false)
  d3.select('#prev-' + title).property("disabled", false)
  d3.select('#next-' + title).property("disabled", pos == game.positions.length - 1)
})

$('#prev-' + title).on('click', function () {
  if (pos > 0) {
    pos = pos - 1
    board.position(game.positions[pos])
  }

  d3.select('#analyze-' + title).property("disabled", false)
  d3.select('#prev-' + title).property("disabled", pos == 0)
  d3.select('#next-' + title).property("disabled", false)
})

$('#reset-' + title).on('click', function() {

  // if pieces have not been moved, reset to beginning of line
  if (!moved) {

    // if looking at a variation, reset to main line
    if (!(variations === undefined) && variations.includes(game.title)) {

      d3.select('#' + game.title)
        .style("box-shadow", null)
        .style("color", null)

      game = data.find(d => d.title === title)
```

```
    pos = game.start - 1

    d3.select('#reset-' + title)
      .classed('btn-primary', false)
      .classed('btn-outline-secondary', true)

  // Otherwise, reset main line (possibly to start of game)
  } else {

    // if reset button is pressed at starting position, totally reset board
    if (pos === game.start - 1) {
      pos = 0
    } else {
      pos = game.start - 1
    }

  }

} else {

  // color reset button correctly if pieces are moved:
  if (!(variations === undefined) && variations.includes(game.title)) {

    d3.select('#reset-' + title)
      .classed('btn-primary', true)
      .classed('btn-success', false)

  } else {

    d3.select('#reset-' + title)
      .classed('btn-outline-secondary', true)
      .classed('btn-success', false)

  }
}

board.position(game.positions[pos])

d3.select('#analyze-' + title).property("disabled", false)
d3.select('#prev-' + title).property("disabled", pos == 0)
d3.select('#next-' + title).property("disabled", pos == game.positions.length - 1)
```

```
  moved = false

})

$('#analyze-' + title).on('click', function() {
  window.open(game.link[pos], '_blank');
})

$('#flip-' + title).on('click', board.flip)


if (!(variations === undefined)) {

  if (!(Array.isArray(variations))) {
    variations = [variations]
  }

  for (let k = 0; k < variations.length; k++) {
    $('#' + variations[k]).on('click', function() {

      // if clicked on active variation, reset to main line
      if (game.title === variations[k]) {

        d3.select('#' + game.title)
          .style("box-shadow", null)
          .style("color", null)

        game = data.find(d => d.title === title)

        // Don't reset position if before start of game
        if (pos >= game.start) {
          pos = game.start - 1
        }

        board.position(game.positions[pos])

        d3.select('#reset-' + title)
          .classed('btn-primary', false)
          .classed('btn-outline-secondary', true)

      // otherwise, set to selected variation
```

```
    } else {

      game = data.find(d => d.title === variations[k])

      // Don't reset position if before start
      // (this allows for the complete viewing of variations)
      if (pos >= game.start) {
        pos = game.start - 1
      }

      board.position(game.positions[pos])

      d3.select('#reset-' + title)
        .classed('btn-outline-secondary', false)
        .classed('btn-success', false)
        .classed('btn-primary', true)

    // need to unset styling for other variations
    variations.filter(name => !(name === variations[k]))
      .map(name =>  {
        d3.select('#' + name)
          .style("box-shadow", null)
          .style("color", null)
          //.style("box-shadow", "inset 0 0 0 0 #2780e3")
          //.style("color", "#2780e3")
      })

    // indicate variation is selected
    d3.select('#' + variations[k])
      .style("box-shadow", "inset 10vw 0 0 0 #2780e3")
      .style("color", "white")

    }

    // activate/deactivate UI
    d3.select('#analyze-' + title).property("disabled", false)
    d3.select('#prev-' + title).property("disabled", pos == 0)
    d3.select('#next-' + title).property("disabled", pos == game.positions.length - 1)

  })
}
```

```
  }

}
```

```
transpose(scandi_t).filter(d => d.type === 'main')
  .map(d => board_setup(transpose(scandi_t), d.title))
```

3.1 Introduction

The Scandinavian Defense begins with 1.e4 d5. Then, white typically captures 2.exd5 and black recaptures with 2…Qxd5. Here we will mainly be looking at at lines following 3.Nc3 Qd8, although there are other options after 3.Nc3 (3…Qa5, 3…Qd6).

3.2 Main Line (2.exd5 Qxd5)

3.2.1 4.d4 Nf6

3.2.1.1 5.Nf3 Bg4

5.Nf3 is the most common response to 4.d4 Nf6. Meet this with 5…Bg4 to put pressure on d4 with the queen.

- 6.Bc4 threatens 7.Bxf7+ then 8.Ne5+
- 6.Be2 placeholder

3.2.1.2 5.Bc4 a6

3.3 What if 2exd4?

3.3.1 2.Nc3 dxe4 3.Nxe4 Qd5

- 4.Nf6 Qd8 transposes back to the mainline
- Instead of retreating, white might protect the knight with 4.Qf4
- 4.Nd3 is also an option (note, black wins a piece if 9.Nxe5)
- 4.d3 is also an option, now black is looking for …e7-e5

3.3.2 2. d4 (The Blackmar-Diemer Gambit)

- temp

3.3.3 2. d3

- The best response to 2. d3 is trading queens, entering a middle-game in which white can not castle

References

Download PGN

References