ROOT Einführung

Statistische Methoden der Datenanalyse

Matthew Beckingham
Henrik Wold Nilsen

29. Oktober 2009

ROOT

- Ein Softwarepaket zur Datenanalyse
- Basiert auf der Programmiersprache C++
- Keine Vorkenntnisse von C++ wird vorausgesetzt für diese Übungsgruppen

- ROOT starten: „root“ und Enter in Konsole eingeben
- Text ausschreiben lassen: cout << ”Test“ << endl;
- Jede Zeile mit Semikolon abschließen
- Logout: .q;

Variablen deklarieren

- Format: Eingabe zur ROOT → Ausgabe von ROOT

- Neue ganze Zahl „x“ deklarieren:
  `int x = 5;`

- Format: *Typ Variablen-Name = Wert;*

- Ausgeben lassen:
  `cout << ”x = “ << x << endl;` → x = 5

- x ändern und ausschreiben lassen:
  `x = 3;`
  `cout << “x ist jetzt = “ << x << endl;` → x ist jetzt = 3

- `int pi_int = 3.14;`
  `cout << “pi_int = “ << pi_int << endl;` → pi_int = 3

- `float pi_float = 3.14;`
  `cout << “pi_float : “ << pi_float << endl;` → pi_float = 3.14

Kommando-Zeile → Script

- Viel Aufwand immer alles in ROOT einzugeben
- Kann auch die ROOT-Kommandos in eine Text-Datei (Script) schreiben, und dann diese Datei an ROOT geben
- Beispiel: Datei mit Name test.C

```
{
  int x = 5;
  int y = 8;
  cout << ”x + y = ” << x+y << endl;
}
```

- Achtung: Text fängt an mit ”{” und hört auf mit ”}”
- 2 Möglichkeiten um test.C mit root zu öffnen:
  - ROOT öffnen mit ”root test.C”
  - Oder: wenn ROOT schon offen ist: ”.x test.C”

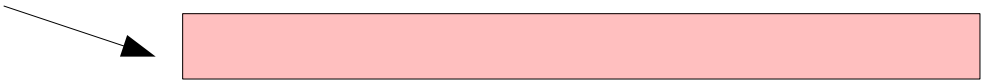
if - Bedingung

- ”if”: wenn ein Kommando ausgeführt werden soll, falls eine Bedingung wahr ist: ”Falls es regnet, nimm einen Regenschirm mit”.

```
int x = 8;
if( x < 10 ) {
  cout << ”x kann man auf zwei Haende zaehlen“ << endl;
}
```

```
x kann man auf zwei Haende zaehlen
```

```
int x = 12;
if( x < 10 ) {
  cout << ”x kann man auf zwei Haende zaehlen“ << endl;
}
```



- Statt

```
if ( Bedingung A ) { etwas; }
if ( nicht Bedingung A ) { was anderes; }
```

geht auch

```
if ( Bedingung A ) { etwas; }
else { was anderes; }
```

”for”- Schleife

- Um ganze Zahlen von 0 bis 10 auszugeben:

```
cout << 0 << endl;
cout << 1 << endl;
        [...]
cout << 10 << endl;
```

- Oder: ”for”-Schleife: ”mach so wie ich sage *N* mal“

```
for( int i=0; i<= 10; i++)  {
  cout << i << endl;
}
```

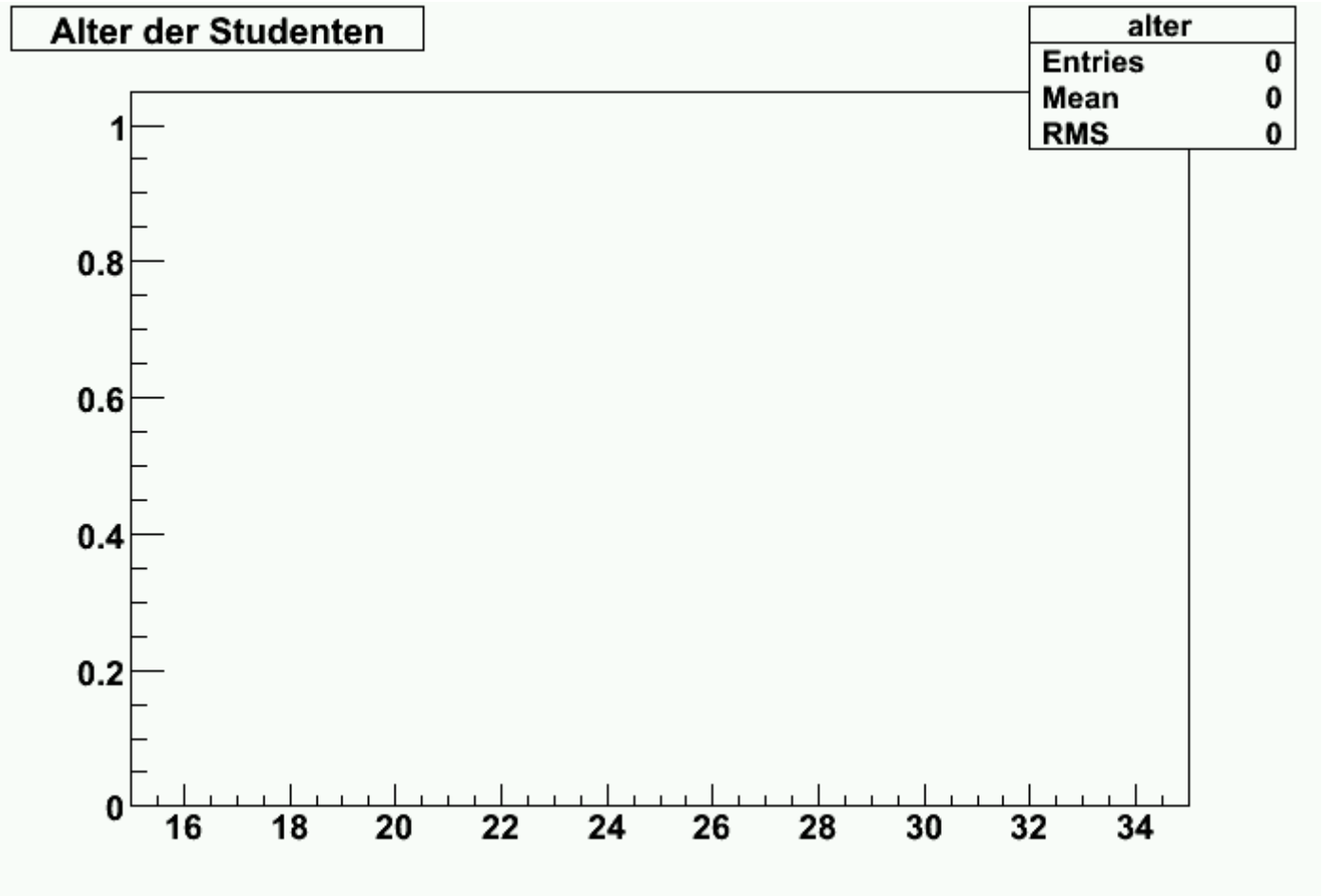
- Hier: ”++” ist eine Abkürzung für ”i = i+1”.
- Andere oft verwendete Abkürzungen:
  - ”x += y” ist gleich ”x = x + y”
  - ”x -= y” ist gleich ”x = x - y”
  - ”x *= y” ist gleich ”x = x * y”
  - ”x /= y” ist gleich ”x = x / y”

Histogramme

- Typ Name = Wert;
- int x = 6;
- Alternativ: Typ Name = Typ(Spezifikation von Eigenschaften)
- int x = int(6);
- ROOT-Histogramm: *Typ* ist ”TH1F“ (eigentlich: *Klasse* statt *Typ*)
  TH1F hist = TH1F(”Name“, ”Titel“, nBins, xLow, xHigh);
- Hier:
  - nBins: Anzahl von Bins im Histogramm – eine ganze Zahl (int)
  - xLow / xHigh : definiert die Intervallgrenzen der x-Achse
- Beispiel:

```
TH1F hist = TH1F( ”alter“, ”Alter der Studenten“, 20, 15, 35);
hist.Draw();
```

Histogramme



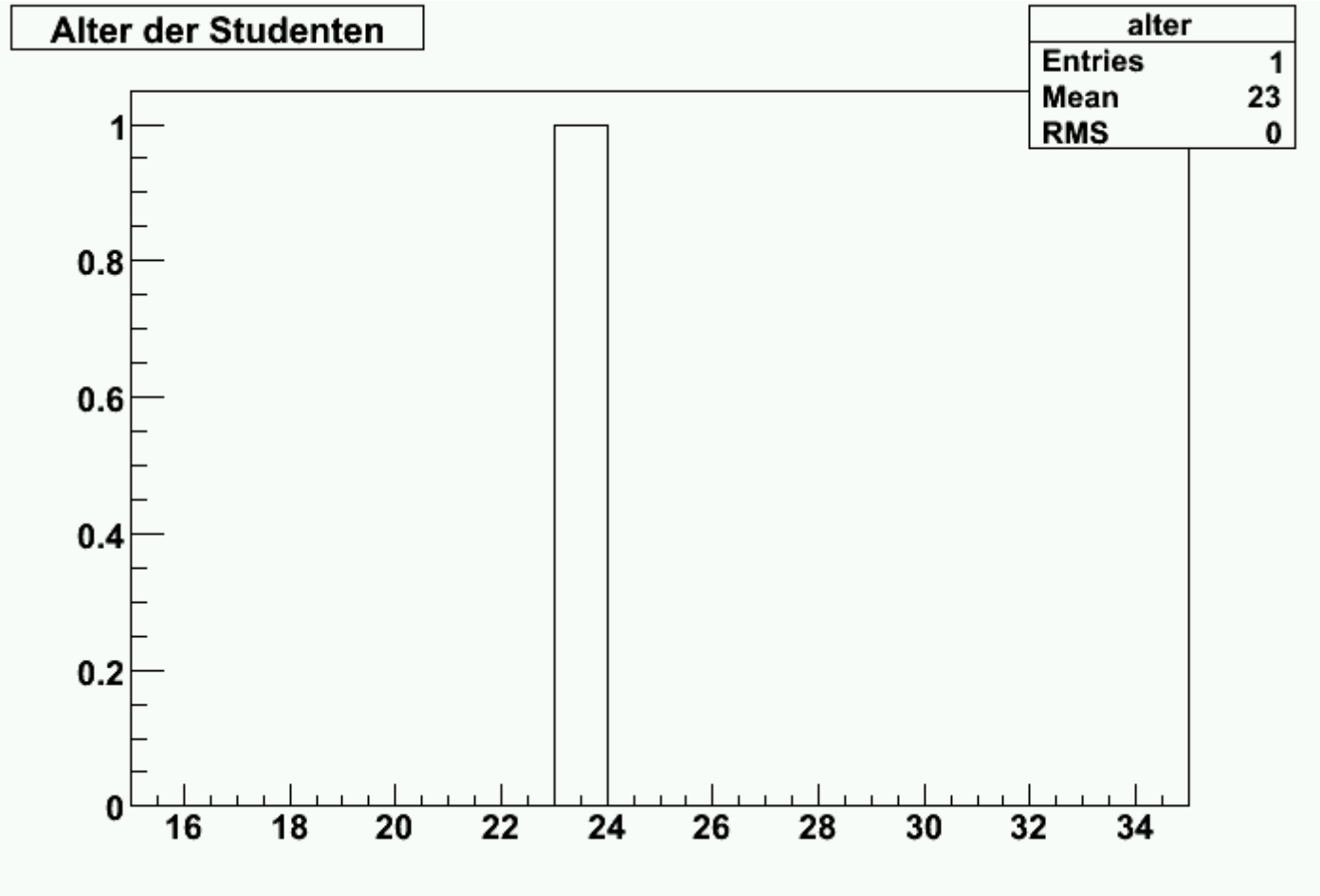

```
TH1F hist = TH1F( ”alter“, ”Alter der Studenten“,  20, 15, 35);
hist.Draw();
```

Histogramme

- ”hist.Draw()” sagt dem Histogramm, dass es sich zeichen soll
- ”hist.Fill(23)” sagt dem Histogramm, dass es die Eintrag ”23” hinzufügen soll

Histogramme

- ”hist.Draw()” sagt dem Histogramm, dass es sich zeichen soll
- ”hist.Fill(23)” sagt dem Histogramm, dass es die Eintrag ”23” hinzufügen soll



Histogramme

- ”hist.Draw()” sagt dem Histogramm, dass es sich zeichen soll
- ”hist.Fill(23)” sagt dem Histogramm, dass es die Eintrag ”23” hinzufügen soll
- ”Draw” und ”Fill” werden ”Mitglied-Funktionen der Histogramm-Klasse TH1F” genannt

Histogramme

- ”hist.Draw()” sagt dem Histogramm, dass es sich zeichen soll
- ”hist.Fill(23)” sagt dem Histogramm, dass es die Eintrag ”23” hinzufügen soll
- ”Draw” und ”Fill” werden ”Mitglied-Funktionen der Histogramm-Klasse TH1F” genannt
- Andere Beispiele für Mitglied-Funktionen von Histogrammen:
  - ”int x = hist.GetBinContent(N)”: jetzt ist der Wert von ”x” gleich der Anzahl Einträge in Bin Nummer N (=1,2,3,...)
  - ”float x = hist.Integral()”: jetzt ist der Wert von ”x” gleich der Anzahl Einträge des Histogramms
  - hist.Scale(x): Multipliziert alle Anzahlen Einträge des Histogramms mit dem Wert von ”x”
  - ”int nBins = hist.GetNbinsX()”: jetzt ist der Wert von ”nBins” gleich der Anzahl Bins des Histogramms

”Zufallszahlen”

- ROOT kann Zufallszahlen ”generieren” durch die Klasse TRandom3 und die Mitglied-Funktionen:
  - Integer(int max): gibt eine zufällige, ganze Zahl zwischen 0 und (max-1)
  - Uniform(): gibt eine zufällige Zahl zwischen 0 und 1

- Beispiel:

```
TRandom3 rnd = TRandom3();
for( int i=0; i<10; i++) { cout << rnd.Integer(10) << endl; }
```

→ 4, 8, 3, 9, 7, 0, 8, 3, 0, 0

- Wenn man ROOT neu startet und das Beispiel wiederholt, kriegt man die gleichen 10 Zahlen nochmal

- Um 10 andere Zahlen zu bekommen, muss die ”Seed” geändert werden durch die Funktion SetSeed(int seed).

```
TRandom3 rnd = TRandom3();
rnd.SetSeed(12);
for( int i=0; i<10; i++) cout << rnd.Integer(10) << endl;
```

→ 1, 9, 3, 9, 1, 3, 0, 0, 8, 6

Mehr Information

- Hier: Links zur zusätzliche Information über ROOT und C++
- Aber: die folgende Informationsquellen enthalten VIEL mehr Information als was man für diese Übungen brauchen.

- C++ programmieren: http://www.cplusplus.com/doc/tutorial/
  - Gründliche Einführung in C++ programmieren

- ROOT: http://root.cern.ch
  - ”Documentation → Reference Guide → Old version 5.18/00”:
  - ”Documentation → User's Guide”: Kleines Buch über ROOT

Aufgabe I

- Bestimmen Sie die Wahrscheinlichkeitsverteilung für der Augenzahlen zweier normale Würfel

Lösungsweg:

- Ein Histogramm erzeugen mit 13 Bins im Intervalle 0.5 bis 13.5
- Zufällige, ganze Zahlen generieren mit Hilfe eines TRandom3 Objektes und der Funktion Integer(6) => Zahlen {0,1,2,3,4,5}
- In eine for-Schleife von 0 bis *N*: zwei Zufallszahlen generieren und jeweils 1 addieren. Die Summe beider Zahlen in die Histogramme eintragen mit Hilfe der Fill-Funktion. Probiere verschiedene Werte für N: 100, 1000, 10000, ...
- Histogramm zeichnen lassen mit Draw()
- Die Histogramme auf 1 normieren mit Hilfe die Funktionen Scale() und Integral(), so dass die relative Verteilung eine Wahrscheinlichkeitsverteilung wird, dann nochmal zeichnen
- Wie wahrscheinlich ist es, eine Summe von z.B. 10 zu bekommen? (GetBinContent(binNr) verwenden)

Aufgabe II – 2D Histogramm

- Ein 2D Histogramm erzeugen mit 15 Bins zwischen 0 und 15 auf der x-Achse, und 15 Bins zwischen 0 und 15 auf der y-Achse:
  - TH2F hist = TH2F(”hist”, ”hist”, 15, 0, 15, 15, 0, 15);
- Die Zahlenpaare (3, 5), (7, 6), (1, 3), (8,8), (6, 4) und (7, 4) eintragen, z.B:
  - hist.Fill(3, 5);
- Zeichnen lassen mit Befehl:
  - hist.Draw(”col”);
- Zeichnen lassen mit Befehl:
  - hist.Draw(”lego”);
- Den Korrelationskoeffizient berechnen und ausgeben lassen:
  - cout << hist.GetCorrelationFactor() << endl;