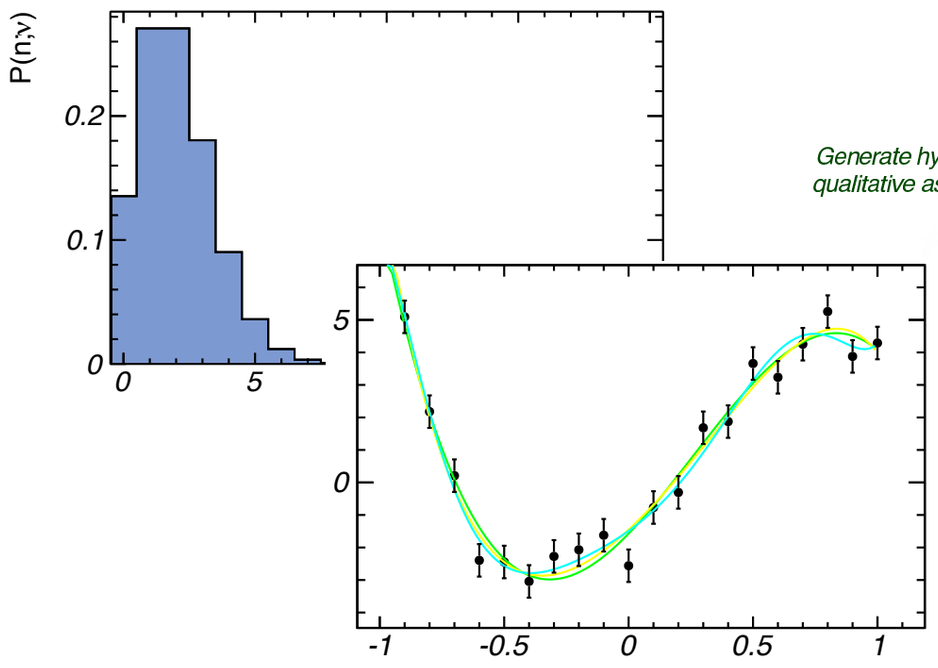
Karlsruher Institut für Technologie

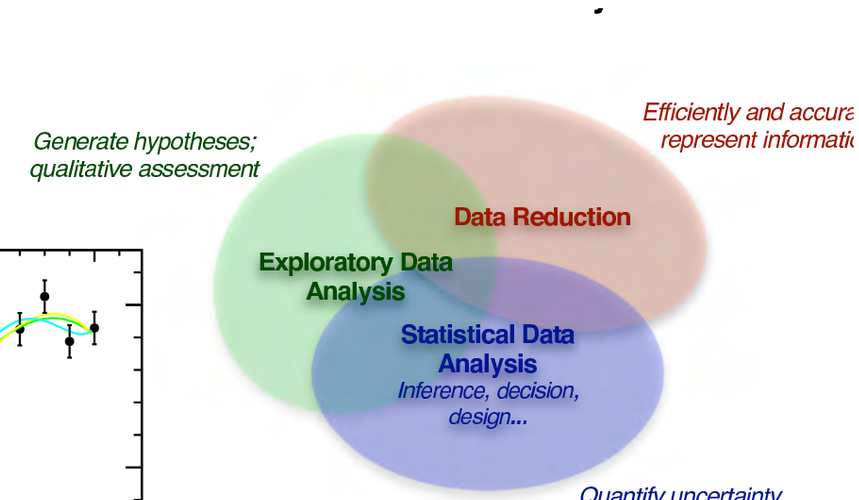
Rechnernutzung in der Physik

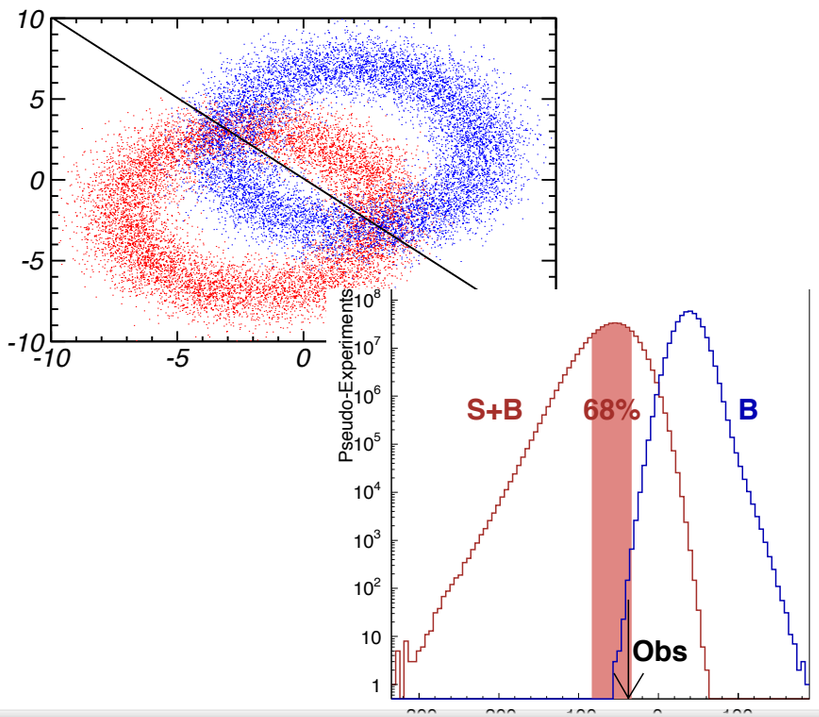
Teil 3 – Statistische Methoden der Datenanalyse

Karlsruher Institut für Technologie
Wintersemester 2012/2013

Ulrich Husemann
Institut für Experimentelle Kernphysik, Karlsruher Institut für Technologie





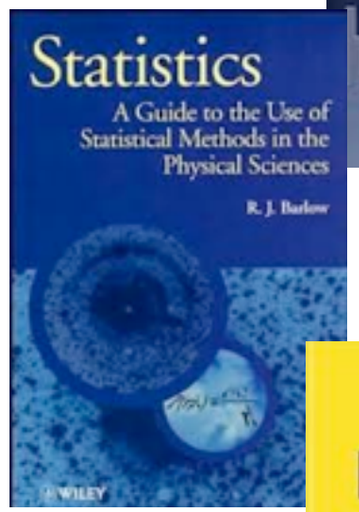


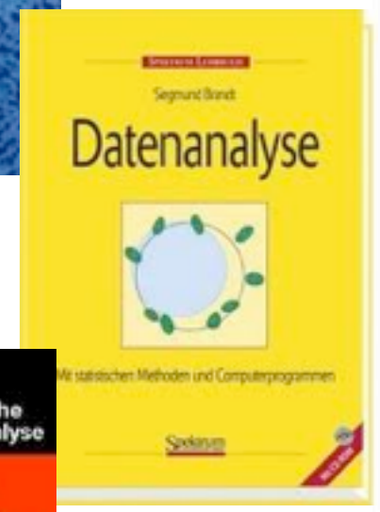
Teil 3 im Überblick

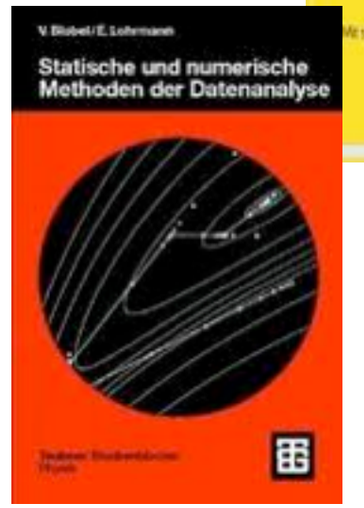
- 3.1 Einführung in das Datenanalyse-Framework ROOT
- 3.2 Grundlagen der Statistik
- 3.3 Beispiele für Wahrscheinlichkeitsverteilungen
- 3.4 Stichproben und Schätzwerte
- 3.5 Monte-Carlo-Methoden
- 3.6 Parameterschätzung
- 3.7 Hypothesentests

Literaturhinweise

- G. Cowan: Statistical Data Analysis, Oxford (1997)
- G. Bohm, G. Zech: Einführung in Statistik und Messwertanalyse für Physiker, DESY E-Buch (2006) http://www-library.desy.de/preparch/books/vstatmp.pdf
- R. J. Barlow: Statistics: A Guide to the Use of Statistical Methods in the Physical Sciences, Wiley (1989)
- S. Brandt: Datenanalyse, Spektrum (1999)
- V. Blobel, E. Lohrmann: Statistische und numerische Methoden der Datenanalyse, Teubner (1998)
- F. James: Statistical Methods in Experimental Physics, World Scientific (2006)







Karlsruher Institut für Technologie

Kapitel 3.1

Einführung in ROOT

Was ist ROOT?

- ROOT
  - Open-Source-Framework zur Datenanalyse (C++-Klassenbibliothek)
  - Entwickelt (hauptsächlich) am CERN, Standard in Teilchenphysik
- ROOT wird benutzt für
  - Verwaltung großer Datenmengen (Trees)
  - Darstellung von Daten (z. B. Histogramme) und Verwaltung von Unsicherheiten
  - Darstellung von mathematischen Funktionen
  - Anpassung von Modellen (Funktionen) an Daten (z. B. χ^2-Methode)
  - Unterstützung von Monte-Carlo-Methoden
  - Graphische Oberflächen mit Darstellung der Daten
  - Entwicklungsumgebung für Datenanalysepakete

Erste Schritte mit ROOT

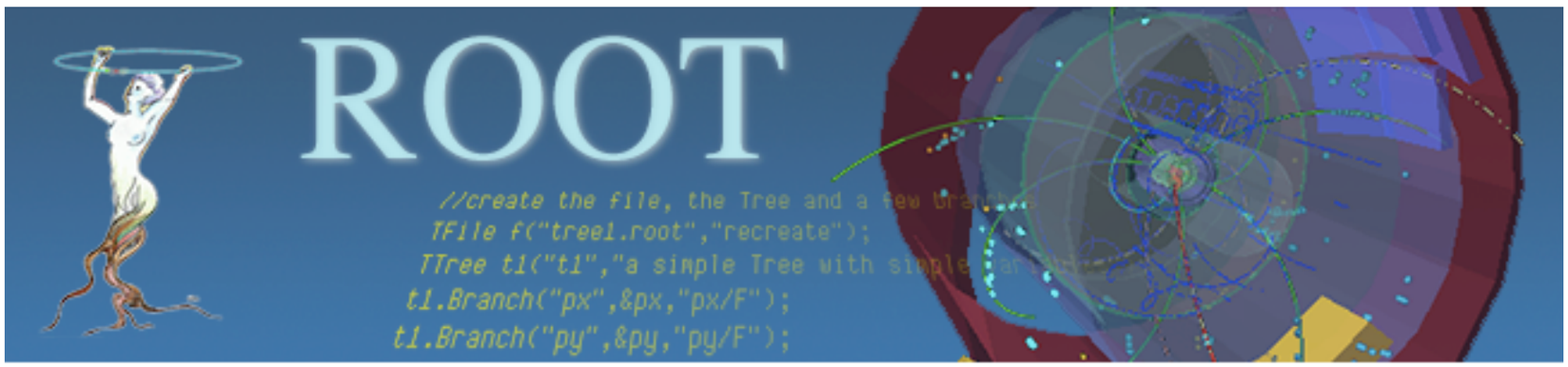
- Starten von ROOT
  - Benötigte Umgebungsvariable: $ROOTSYS
    → $ROOTSYS/bin in $PATH, $ROOTSYS/lib in $LD_LIBRARY_PATH
  - In der Shell

```
prompt> root
  ******************************************
  *                                        *
  *        W E L C O M E  to  R O O T      *
  *                                        *
  *   Version   5.34/00       5 June 2012  *
  *                                        *
  *  You are welcome to visit our Web site *
  *          http://root.cern.ch           *
  *                                        *
  ******************************************

ROOT 5.34/00 (branches/v5-34-00-patches@44555, Jun 05 2012, 16:18:52 on macosx64)

CINT/ROOT C/C++ Interpreter version 5.18.00, July 2, 2010
Type ? for help. Commands must be C++ statements.
Enclose multiple statements between { }.
root [0]
```

- Beenden: `root [1] .q`

Dokumentation

- Webseite: http://root.cern.ch
  - Benutzerhandbuch: http://root.cern.ch/drupal/content/users-guide
  - Aktuelle Klassenreferenz: http://root.cern.ch/root/html534/ClassIndex.html
- Viele Einführungen im Internet, z. B.
  - http://www.ekp.kit.edu/~quast/Skripte/diving_into_ROOT.pdf
  - http://www.nevis.columbia.edu/~seligman/root-class/



Benutzung

- Kommandozeile
  - C++-Interpreter (CINT)
  - C++: starke Typisierung → manchmal umständlich auf der Kommandozeile
  - Lösung: Bindings für moderne Skriptsprachen, besonders Python
- Makros
  - Kurze C++-Funktionen oder -Programme von CINT aus aufrufbar
  - Vorteil: Ausführung in CINT→ kurze Entwicklungszyklen
- Kompilierte Programme
  - Klassisch (Compiler, Linker, Makefile, …) oder auf ROOT-Kommandozeile
  - Vorteil: schnellere Ausführung, strengere Fehlerkontrolle

ROOT-Makros: Beispiel

- Das Makro (Dateiname: macro_01.cxx)

```
void macro_01()
{
  cout << "Die ersten 10 Quadratzahlen:" << endl;
  for( int i = 1; i <= 10; ++i ) {
    cout << i*i << endl;
  }
}
```

- Ausführung (Dateiname = Macroname)

```
root[0] .x macro_01.cxx
```

- Laden und Ausführen beliebiger Funktionen

```
root[0] .L macro_01.cxx
root[1] macro_01()
```

Kompiliertes ROOT-Makro

- Macro erweitert um Include-Dateien, Namespace usw.

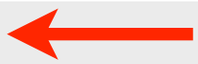
```
#include <iostream>
using namespace std;

void macro_01()
{
  cout << "Die ersten 10 Quadratzahlen:" << endl;
  for( int i = 1; i <= 10; ++i ) {
    cout << i*i << endl;
  }
}
```

- Automatisches Kompilieren in CINT

```
root[0] .x macro_01.cxx+
```

bzw.

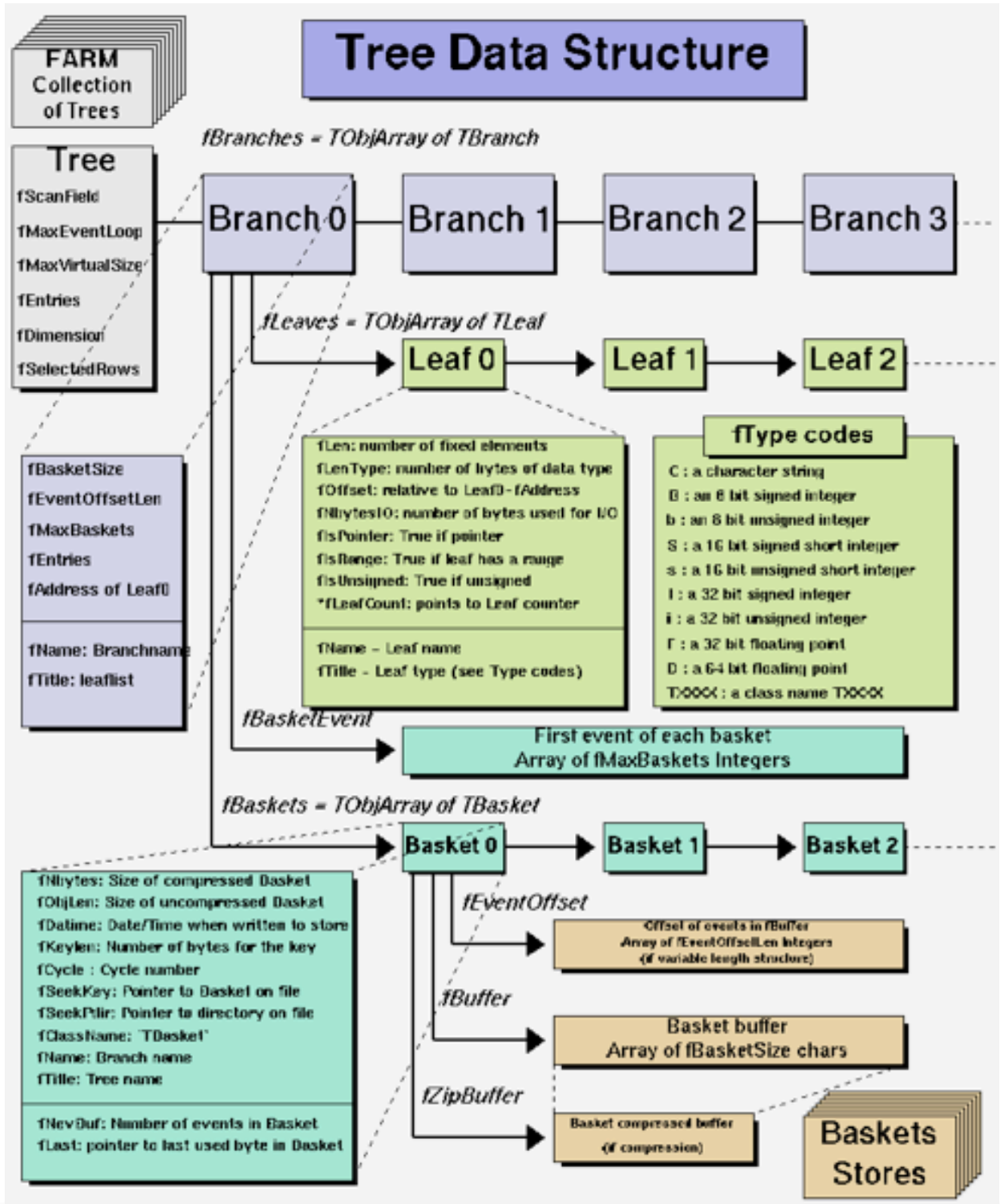
```
root[0] .L macro_01.cxx+
root[1] macro_01()
```



Datenmodell: ROOT Trees

- Tree
  - Datenspeicherung in Klassen (z. B. alle Objekte einer Teilchenkollision in einem „Ereignis”)
  - Hierarchie: Tree – Branch – Leaf
- Einfachster Tree: n-Tupel
  - Tree aus identischen Variablentypen (z. B. float)
  - Beispiel: Viererimpulse

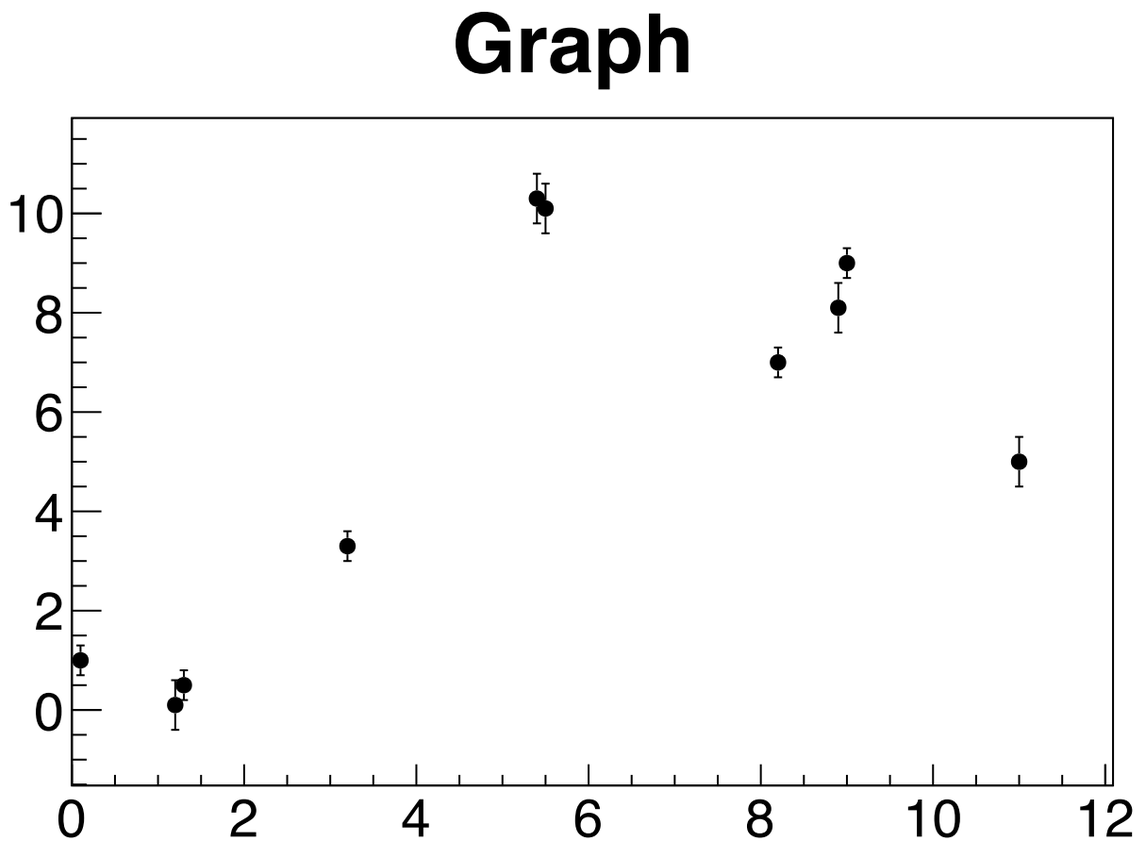
```
root[0] TNtuple* nt = new TNtuple( "nt",
"Mein n-Tupel", "E:px:py:pz" )
root[1] nt->Fill( 10.0, 9.0, 0.5, 0 )
root[2] nt->Fill( 25.3, -5.0, 20.0, -0.5 )
…
```



Graphen

- Darstellung von Messdaten
  - Datenpunkt (x,y)
  - Unsicherheiten („Fehlerbalken”), ggf. asymmetrisch

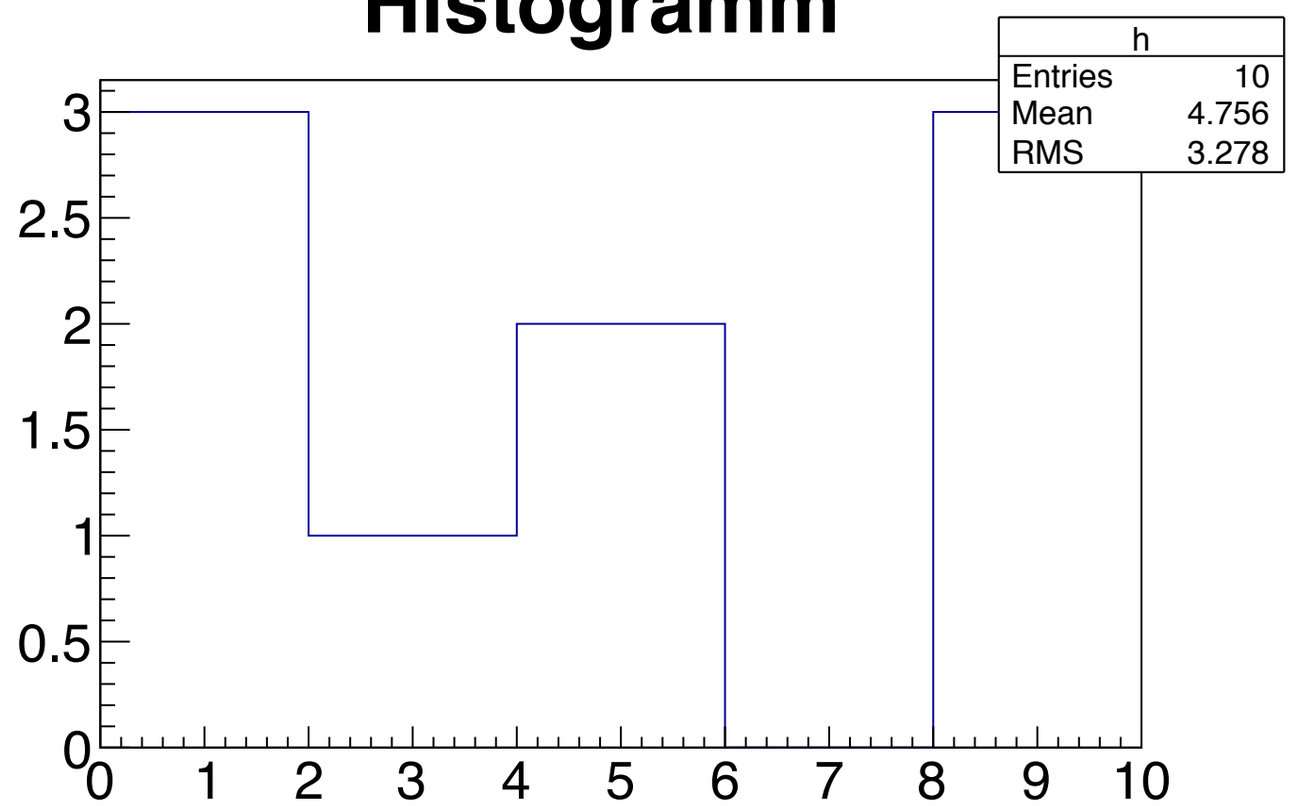
```
void graph()
{
  const int N = 10;
  float x[N] = { 1.3, 5.4, 3.2, 5.5, 8.2,
                 8.9, 9.0, 1.2, 0.1, 11 };
  float y[N] = { 0.5, 10.3, 3.3, 10.1, 7.0,
                 8.1, 9.0, 0.1, 1.0, 5.0 };
  float error[N] = { 0.3, 0.5, 0.3, 0.5, 0.3,
                     0.5, 0.3, 0.5, 0.3, 0.5 };

  TGraphErrors* g =
    new TGraphErrors( N, x, y, 0, error );
  g->SetMarkerStyle( 20 );
  g->SetMarkerSize( 1.0 );
  g->Draw( "ap" );
}
```



Histogramme

- Häufigkeitsverteilung („Histogramm”)
  - Kompakte Darstellung großer Datenmengen
  - Einteilung eines Intervalls in Teilintervalle („Bins”) gleicher oder variabler Breite
  - Inhalt des Bins: Anzahl der Ereignisse im Teilintervall
- Bemerkungen:
  - Auch in >1 Dimensionen
  - Es gibt Over-/Underflow-Bins
  - Statistikbox: Einträge, Mittelwerte usw.

```
void histogramm()
{
  TH1F* h = new TH1F( "h", "Histogramm", 5, 0, 10 );
  float data[10] = { 1.3, 5.4, 3.2, 5.5, 8.2,
                     8.9, 9.0, 1.2, 0.1, 11.0 };
  for( int i = 0; i < 10; ++i ) {
    h->Fill( data[i] );
  }
  h->Draw();
}
```

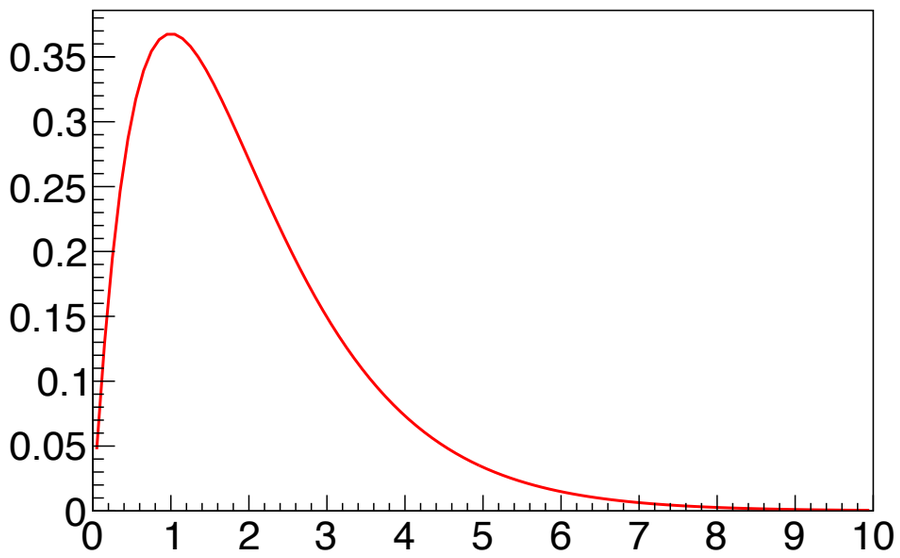


Funktionen

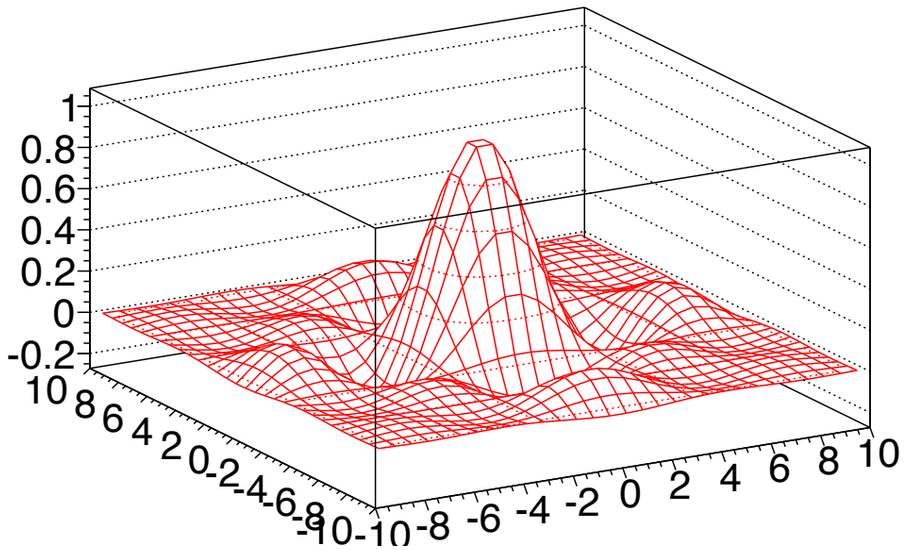
- Standardfunktionen vordefiniert
- Einfache Funktionen über Strings definierbar
- Komplizierte Funktionen über C++-Funktionen mit festem Interface

```
void funktion()
{
  TF1* f1 = new TF1( "f1", "exp(-x)*x", 0, 10 );
  f1->Draw();

  TF2* f2 =
    new TF2( "f2", func2, -10, 10, -10, 10, 0 );
  f2->Draw( "surf" );
}

double func2( double* val, double* par )
{
  double x = val[0];
  double y = val[1];
  return sin(x)*sin(y)/(x*y);
}
```

exp(-x)*x



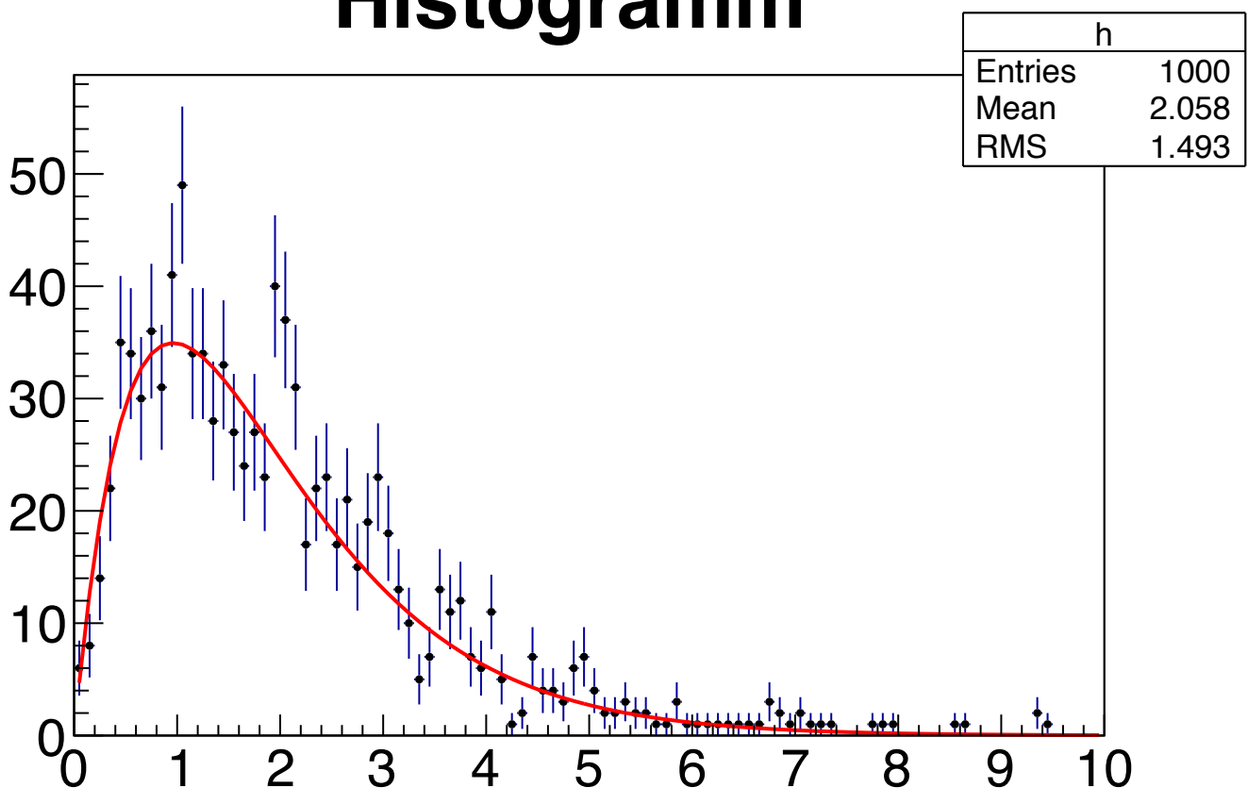
sin(x)*sin(y)/(x*y)



Anpassungsrechnung („Fit”)

- Erzeugung von Zufallsverteilung mit Monte-Carlo-Methode
- Anpassung mit parametrisierter Funktion

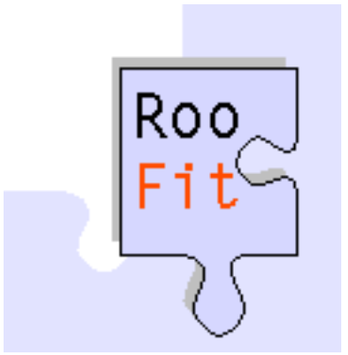
```
void anpassung()
{
  TH1* h =
    new TH1F( "h", "Histogramm", 100, 0, 10 );
  TF1* f1 =
    new TF1( "f1", "exp(-x)*x", 0, 10 );

  h->FillRandom( "f1", 1000 );
  h->SetMarkerStyle( 20 );
  h->SetMarkerSize( 0.5 );

  TF1* ffit =
    new TF1( "ffit", "[0]*exp(-[1]*x)*x", 0, 10 );
  ffit->SetParameters( 10, 1 );
  h->Fit("ffit","","e");
}
```



Grundlagen zu MC-Methode und Parameteranpassung → diese Vorlesung

ROOT-basierte Analysepakete

- ROOT-Klassenbibliothek: Grundlage für moderne Datenanalysepakete
- RooFit: Datenmodellierung
  - Ursprünglich für BaBar-Experiment entwickelt
  - Komplizierte Anpassungen, Toy-Monte-Carlo



- RooStats: statistische Datenanalyse
  - Basierend auf ROOT und RooFit
  - Entwickelt sich zum Standard-Werkzeug am LHC für Parameterschätzung, Konfidenzintervalle, Hypothesentests, Kombination von Analysen, …
- TMVA: Multivariate Klassifikation von Daten
  - Paket für multivariate Regressionsmethoden
  - Likelihood-Methoden, neuronale Netze, …

Zusammenfassung

- ROOT: C++Framework zur Datenanalyse (Quasistandard)
- Speicherung von Daten: ROOT Trees
- Darstellung von Daten: Graphen, Histogramme, Funktionen
- Anpassung von Daten („Fit”)
- Auf ROOT basierende Analysepakete, z. B. RooFit, TMVA

Karlsruher Institut für Technologie

Kapitel 3.2

Grundlagen der Statistik

Vorhersagbar vs. zufällig

- Klassische Physik:
  - Gleichungen deterministisch: Ergebnis exakt vorhersagbar
  - Beispiele: Pendel, Planetenbahnen, Elektromagnetismus, …
  - Aber: manchmal extreme Abhängigkeit von Anfangsbedingungen (z. B. Dreikörperproblem, Lottozahlen)
  - Beschreibung von Vielkörpersystemen über statistische Eigenschaften (z. B. Bewegung der Gasmoleküle → Temperatur eines Gases)
- Quantenphysik:
  - Schrödingergleichung determistisch
  - Kopenhagener Deutung: Quadrat der Wellenfunktion = Wahrscheinlichkeitsdichte
  - Nur statistische Aussagen über Observablen, z. B. Erwartungswert des Aufenthaltsort, mittlere Lebensdauer

Messen und Statistik

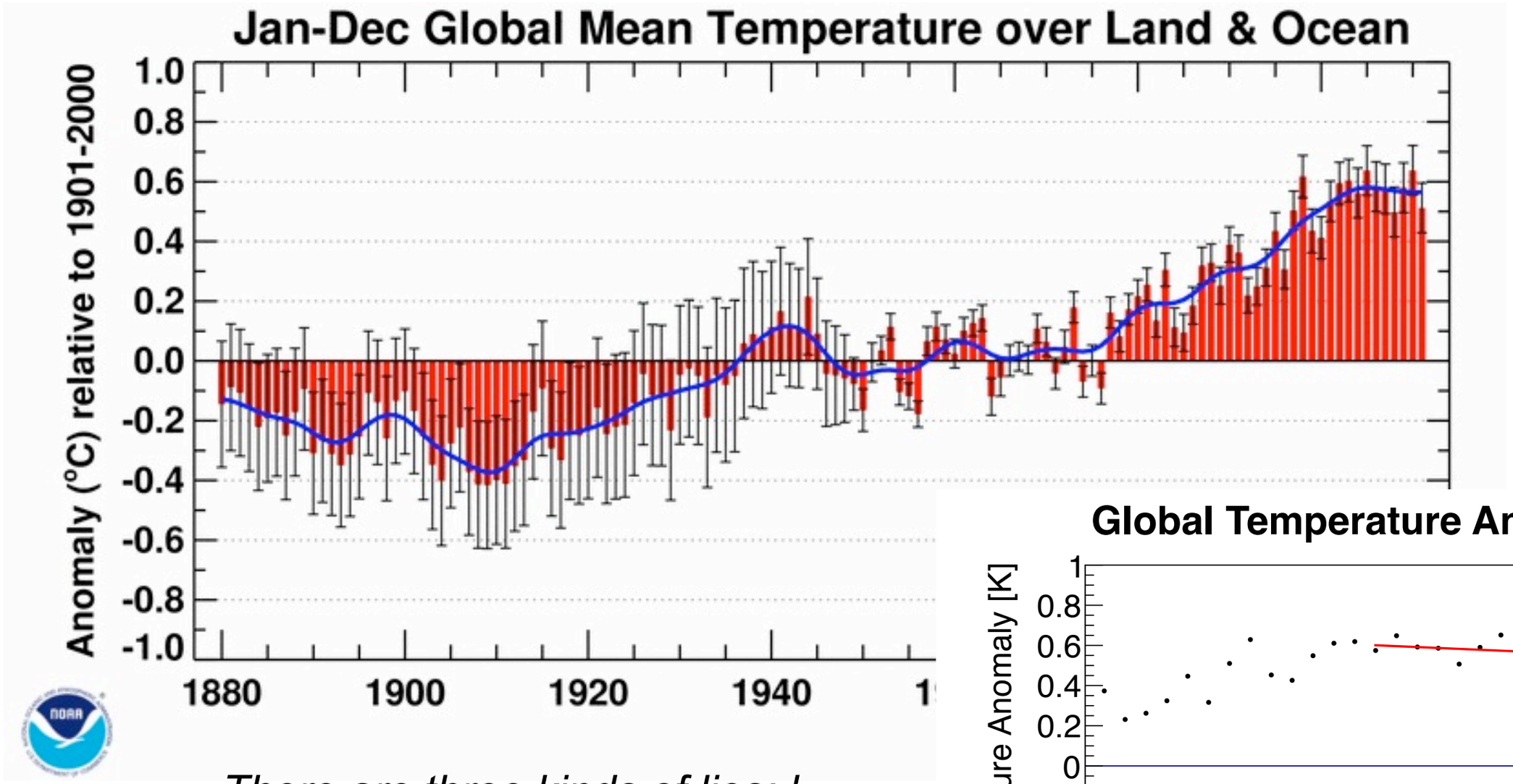
- Auswertung von Messreihen:
  - Schätzung von Parametern aus endlicher Stichprobe (z.B. Mittelwert)
  - Verteilung der Messwerte um den Mittelwert
  - Anpassung von Funktionen (z.B. „Ausgleichsgerade”)
  - Kombination von Messungen

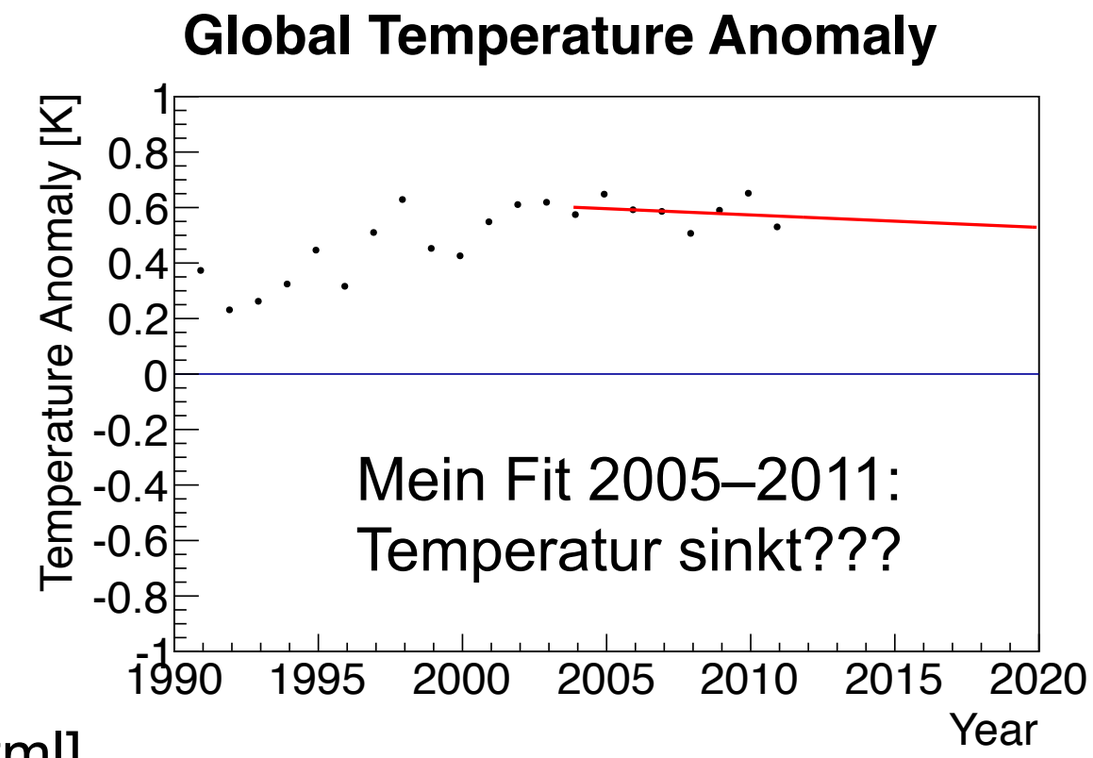
- Unsicherheit von Messgrößen:
  - Einfluss schwer kontrollierbarer Größen (oft: „Messfehler”, besser: „Messunsicherheit”) mit Methoden der Statistik behandelt (z. B. Ablesegenauigkeit, Rauschen)
  - Angabe von Messwert ohne Unsicherheit ist sinnlos!
  - Statistische und systematische Unsicherheiten, z.B. Masse des Top-Quarks: $m_t = 173.2 \pm 0.6\ (\text{stat.}) \pm 0.8\ (\text{syst.})\ \text{GeV}/c^2$

Statistische Methoden

- Sammlung und Analyse von Daten
  - Forschung in Naturwissenschaften, Medizin
  - Finanzwelt: Börsendaten, Wechselkurse, …
  - Data-Mining in der Wirtschaft: Google, Payback-Karten, …
- Test von Hypothesen, Klassifizierung von Daten, Bewertung von Risiken
  - Ist das 2012 am LHC entdeckte Teilchen wirklich das Higgs-Boson?
  - Ist diese E-Mail Spam?
  - Gibt es eine globale Erwärmung?
- Als Naturwissenschaftler sollten Sie solche Vorgänge verstehen und bewerten können!

Beispiel: Klimaforschung

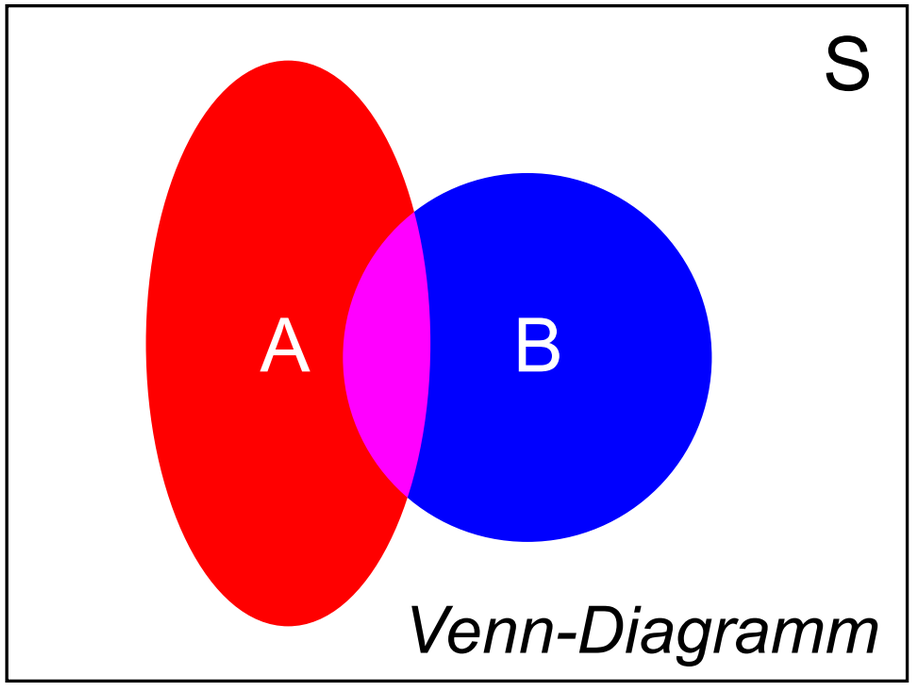




There are three kinds of lies: l ies, damned lies, and statistics
(zitiert von Mark Twain, angeblich von Benjamin Disraeli)

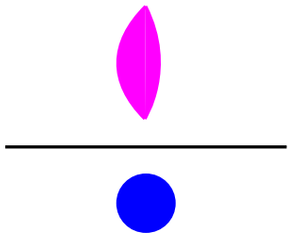
[http://www.ncdc.noaa.gov/cmb-faq/anomalies.html]

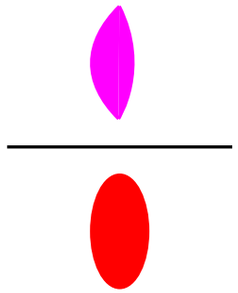
Satz von Bayes in Bildern

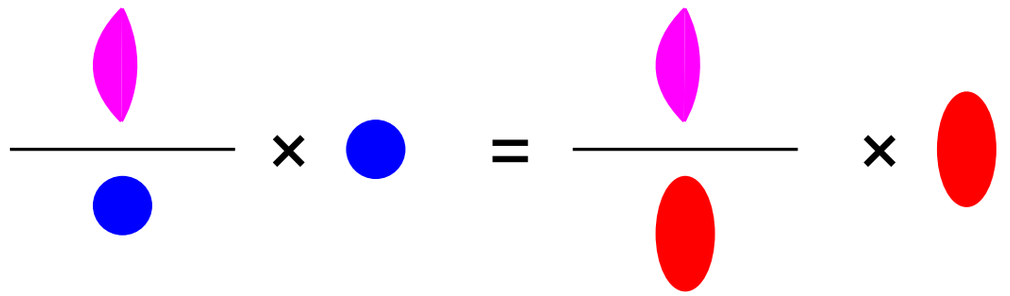


$P(\mathrm{A}) =$ 

$P(\mathrm{B}) =$ 

$P(\mathrm{A}|\mathrm{B}) =$ 

$P(\mathrm{B}|\mathrm{A}) =$ 



$$P(\mathrm{A}|\mathrm{B}) \times P(\mathrm{B}) = P(\mathrm{B}|\mathrm{A}) \times P(\mathrm{A})$$

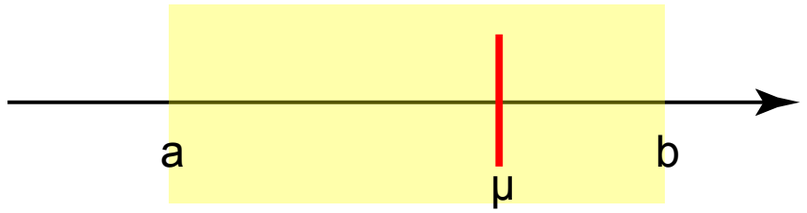
Thomas Bayes
(1702–1761)

Fernsehquiz („Ziegenproblem”)

- Spielidee (“Let’s Make a Deal”, „Geh aufs Ganze”)
  - Drei Türen, hinter einer Tür ist ein Auto, hinter zwei Türen nur eine Ziege
  - Gast wählt eine Tür aus (z. B. 1), diese bleibt aber geschlossen
  - Moderator öffnet eine andere Tür (z. B. 3)
  - Gast muss entscheiden: bei Tür 1 bleiben oder auf Tür 2 wechseln?

Denkschulen der Statistik

- (klassische) frequentistische Statistik
  - Wahrer Wert μ einer Größe existiert (ist aber unbekannt), wird im Limes unendliche vieler Messungen erreicht: P(A) = lim …
  - Typisches Resultat: Fehlerintervall → „68% aller aus Daten gebildeten Intervalle [a,b] erhalten den wahren Wert μ”



- Bayes’sche Statistik: subjektive Wahrscheinlichkeit
  - Wahrer Wert einer Größe existiert nicht
  - P(A) = Grad der Sicherheit (“degree of belief”), dass Ereignis A eintritt
  - Typisches Resultat: Fehlerintervall → „Mittelwert liegt mit 68% Wahrscheinlichkeit in Intervall [a,b]”
  - Benutzung von Vorwissen („Prior”) erlaubt:

$$P(\text{Theorie}|\text{Daten}) \sim P(\text{Daten}|\text{Theorie}) \times P(\text{Theorie})$$