In [1]:

```
%matplotlib inline

import warnings
import numpy as np
import pandas as pd
import scipy.stats as st
import statsmodels as sm
import matplotlib
import matplotlib.pyplot as plt

matplotlib.rcParams['figure.figsize'] = (16.0, 12.0)
matplotlib.style.use('ggplot')
```

In [2]:

```
# Models from data creating
def best_fit_distribution(data, bins=200, ax=None):

    # Plot histogram of original dataset
    y, x = np.histogram(data, bins=bins, density=True)
    x = (x + np.roll(x, -1))[:-1] / 2.0

    # Distributions to check
    DISTRIBUTIONS = [
        st.alpha,st.anglit,st.arcsine,st.beta,st.betaprime,st.bradford,st.burr,st.cauchy,st.chi,st.chi2,st.cosine,
        st.dgamma,st.dweibull,st.erlang,st.expon,st.exponnorm,st.exponweib,st.exponpow,st.f,st.fatiguelife,st.fisk,
        st.foldcauchy,st.foldnorm,st.frechet_r,st.frechet_l,st.genlogistic,st.genpareto,st.gennorm,st.genexpon,
        st.genextreme,st.gausshyper,st.gamma,st.gengamma,st.genhalflogistic,st.gilbrat,st.gompertz,st.gumbel_r,
        st.gumbel_l,st.halfcauchy,st.halflogistic,st.halfnorm,st.halfgennorm,st.hypsecant,st.invgamma,st.invgauss,
        st.invweibull,st.johnsonsb,st.johnsonsu,st.ksone,st.kstwobign,st.laplace,st.levy,st.levy_l,st.levy_stable,
        st.logistic,st.loggamma,st.loglaplace,st.lognorm,st.lomax,st.maxwell,st.mielke,st.nakagami,st.ncx2,st.ncf,
        st.nct,st.norm,st.pareto,st.pearson3,st.powerlaw,st.powerlognorm,st.powernorm,st.rdist,st.reciprocal,
        st.rayleigh,st.rice,st.recipinvgauss,st.semicircular,st.t,st.triang,st.truncexpon,st.truncnorm,st.tukeylambda,
        st.uniform,st.vonmises,st.vonmises_line,st.wald,st.weibull_min,st.weibull_max,st.wrapcauchy
    ]

    best_distribution = st.norm
    best_params = (0.0, 1.0)
    best_sse = np.inf

    # Distribution parameters
    for distribution in DISTRIBUTIONS:

        # Try to fit the distribution
        try:

            with warnings.catch_warnings():
```

```
                warnings.filterwarnings('ignore')

                # fit dist to data
                params = distribution.fit(data)


                arg = params[:-2]
                loc = params[-2]
                scale = params[-1]

                # Calculate fitted pdf
                pdf = distribution.pdf(x, loc=loc, scale=scale, *arg)
                sse = np.sum(np.power(y - pdf, 2.0))

                # axis work
                try:
                    if ax:
                        pd.Series(pdf, x).plot(ax=ax)
                    end
                except Exception:
                    pass

                # proof of perfect distribution
                if best_sse > sse > 0:
                    best_distribution = distribution
                    best_params = params
                    best_sse = sse

        except Exception:
            pass

    return (best_distribution.name, best_params)
```

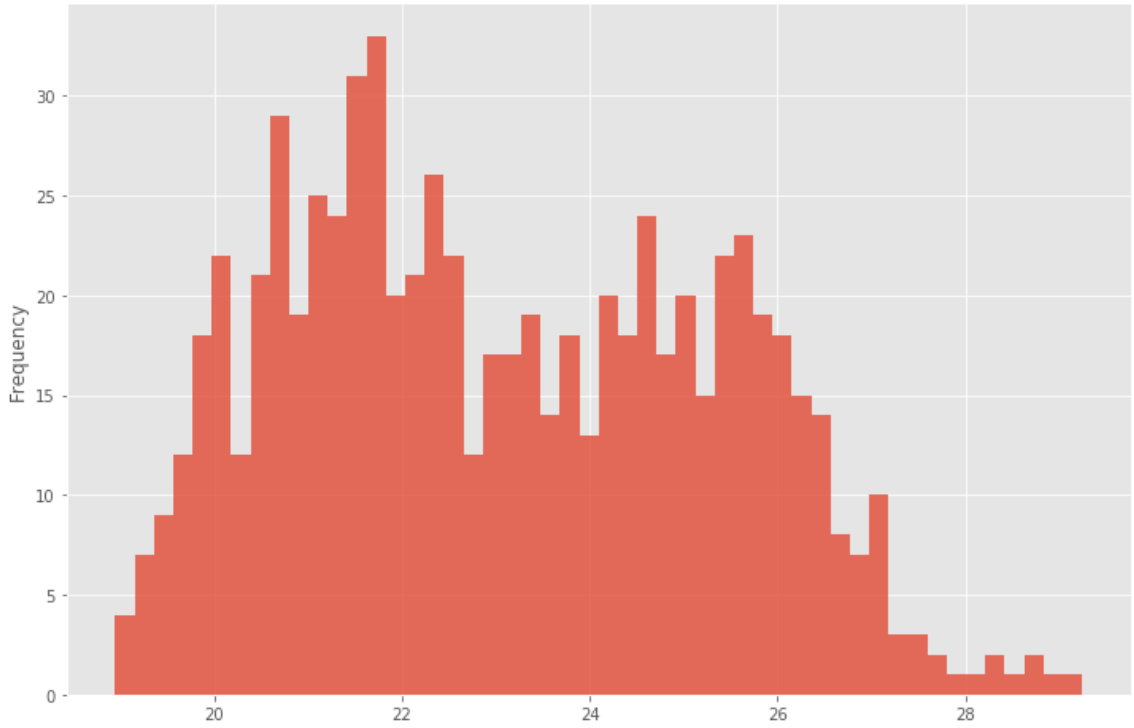
In [5]:

```
#dataset inplace. Import from statsmodels library

data = pd.Series(sm.datasets.elnino.load_pandas().data.set_index('YEAR').values.
ravel())
```

In [12]:

```
# normed hist - non
plt.figure(figsize=(12,8))
ax = data.plot(kind='hist', bins=50, normed=False, alpha=0.8);
```

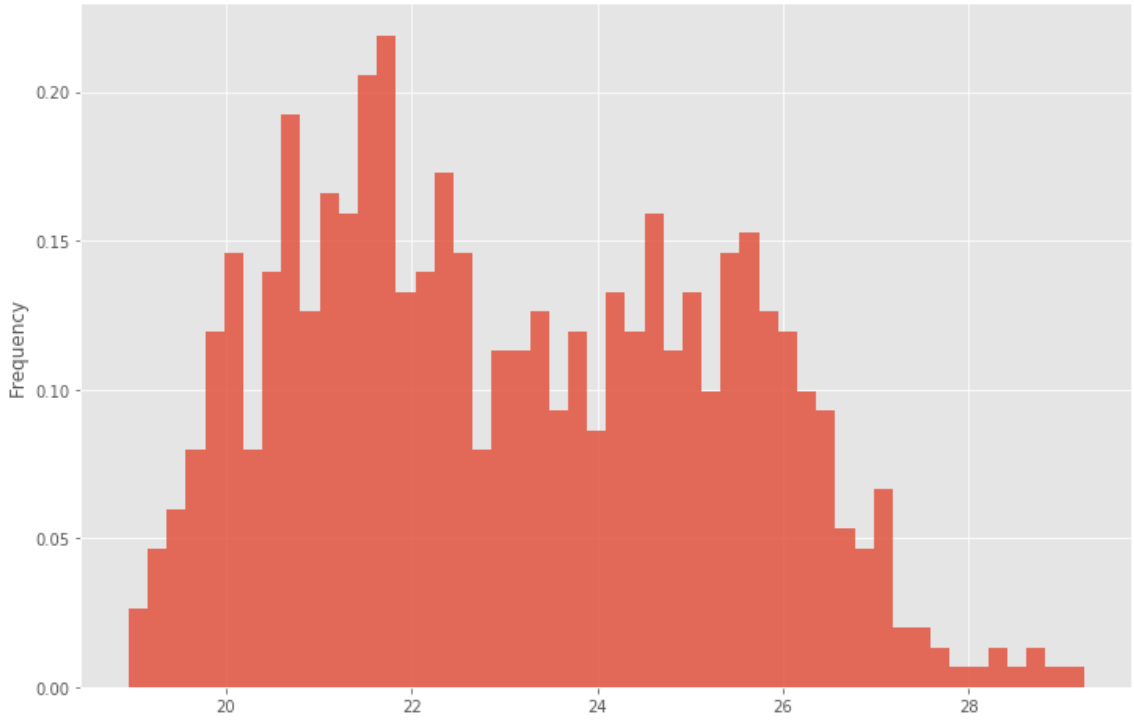
```
/home/evgen/anaconda3/lib/python3.5/site-packages/matplotlib/axes/_a
xes.py:6462: UserWarning: The 'normed' kwarg is deprecated, and has
been replaced by the 'density' kwarg.
  warnings.warn("The 'normed' kwarg is deprecated, and has been "
```



In [10]:

```
# normed hist - True
plt.figure(figsize=(12,8))
ax = data.plot(kind='hist', bins=50, normed=True, alpha=0.8);
```

```
/home/evgen/anaconda3/lib/python3.5/site-packages/matplotlib/axes/_a
xes.py:6462: UserWarning: The 'normed' kwarg is deprecated, and has
been replaced by the 'density' kwarg.
  warnings.warn("The 'normed' kwarg is deprecated, and has been "
```




In [13]:

```
def make_pdf(dist, params, size=10000):


    # Separate param-rs
    arg = params[:-2]
    loc = params[-2]
    scale = params[-1]


    start = dist.ppf(0.01, *arg, loc=loc, scale=scale) if arg else dist.ppf(0.01
, loc=loc, scale=scale)
    end = dist.ppf(0.99, *arg, loc=loc, scale=scale) if arg else dist.ppf(0.99,
loc=loc, scale=scale)

    # Build pdf
    x = np.linspace(start, end, size)
    y = dist.pdf(x, loc=loc, scale=scale, *arg)
    pdf = pd.Series(y, x)

    return pdf
```

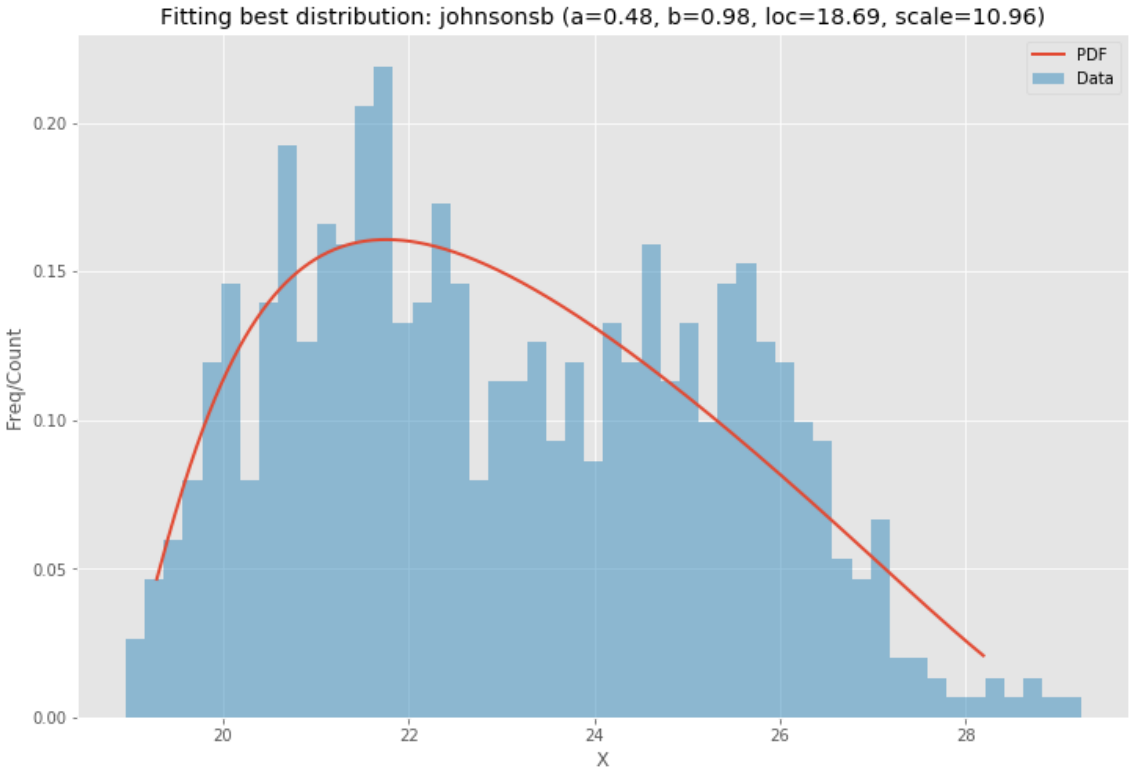
In [14]:

```
# Find best fit distribution
best_fit_name, best_fir_paramms = best_fit_distribution(data, 200, ax)
best_dist = getattr(st, best_fit_name)
```

In [27]:

```
# Make pdf
pdf = make_pdf(best_dist, best_fir_paramms)
print('Best fit-distribution name: ', best_fit_name)
print('Best paramms of this distribution: ', best_fir_paramms)

# Display
plt.figure(figsize=(12,8));
ax = pdf.plot(lw=2, label='PDF', legend=True)
data.plot(kind='hist', bins=50, normed=True, alpha=0.5, label='Data', legend=Tru
e, ax=ax)

param_names = (best_dist.shapes + ', loc, scale').split(', ') if best_dist.shape
s else ['loc', 'scale']
param_str = ', '.join(['{}={:0.2f}'.format(k,v) for k,v in zip(param_names, best
_fir_paramms)])
dist_str = '{} ({})'.format(best_fit_name, param_str)


plt.title('Fitting best distribution: '+dist_str);
plt.xlabel('X');
plt.ylabel('Freq/Count');
```

```
Best fit-distribution name:  johnsonsb
Best paramms of this distribution:  (0.48180802182650617, 0.98358265
71292925, 18.689276660398363, 10.958573456318526)

/home/evgen/anaconda3/lib/python3.5/site-packages/matplotlib/axes/_a
xes.py:6462: UserWarning: The 'normed' kwarg is deprecated, and has
been replaced by the 'density' kwarg.
  warnings.warn("The 'normed' kwarg is deprecated, and has been "
```



In [24]:

Out[24]:

```
'johnsonsb(a=0.48, b=0.98, loc=18.69, scale=10.96)'
```

In [17]:

Out[17]:

```
(0.48180802182650617,
 0.9835826571292925,
 18.689276660398363,
 10.958573456318526)
```

In [18]:

Out[18]:

```
<function __main__.best_fit_distribution>
```

In [19]:

Out[19]:

```
<scipy.stats._continuous_distns.johnsonsb_gen at 0x7fcf70ccecf8>
```